Click to verify

Hello everyone,As you can see in the codepen just below : different events are displayed correctly.But, if i had this simple code :views: { dayGridMonth: { eventLimit: 3, }, },I obtain this result :As you can see, the 12th of August, i have only 2 events which are displayed, not 3 !And, if i change it to :views: { dayGridMonth: { eventLimit: 1, }, },I have
this display :I have '+1 more' everyday !Is there a solution ? Is there a parameter i don't know ?Thank you for your time ! edit doc v6 v5 v4 v3 v2 v1 In, dayGrid view, the max number of events within a given day, not counting the +more link. The rest will show up in a popover. Just like with dayMaxEventRows, if true is specified, the number of events
will be limited to the height of the day cell. For timeGrid and timeline views, see eventMaxStack. You cant perform that action at this time. Limits the number of events displayed on a day. The rest will show up in a popover. Boolean, Integer. default: false When there are too many events, a link that looks like +2 more is displayed. The exact action
that happens when the user clicks the link is determined by eventLimitClick. A value of false (the default) will display all events as-is. A value of true will limit the number of events to the height of the day cell. View a live demo. An integer value will limit the events to a specific number of rows. For the all-day section in the TimeGrid views, a value of
true will limit the number of events to 5. To fine-tune this, use View-Specific Options like this: var calendar = new Calendar(calendarEl, { eventLimit: true, // for all non-TimeGrid views views: { timeGrid: { eventLimit: 6 // adjust to 6 only for timeGridWeek/timeGridDay } }}); Limits the number of events displayed on a day. The rest will show up in a
popover. Boolean, Integer. default: false When there are too many events, a link that looks like +2 more is displayed. The exact action that happens when the user clicks the link is determined by eventLimitClick. A value of false (the default) will display all events as-is. A value of true will limit the number of events to the height of the day cell. View a
live demo. An integer value will limit the events to a specific number of rows. For the all-day section in the agenda views, a value of true will limit the number of events to 5. To fine-tune this, use View-Specific Options like this: $('#calendar').fullCalendar({ eventLimit: true, // for all non-agenda views views: { agenda: { eventLimit: 6 // adjust to 6 only
for agendaWeek/agendaDay } }}); Determines the text of the link created by the eventLimit setting. String, Function. default: "more" If a string is specified, the resulting text will be prepended with the number of events being obscured, like +2, resulting in something like +2 more. If a function is specified, the function will be given a single argument,
the number of events being obscured, and must return the final text string to be used. The default value of this option is affected by the current locale. A value of false (the default) will display all events as-is.,Limits the number of events displayed on a day. The rest will show up in a popover.,An integer value will limit the events to a specific number of
rows.,A value of true will limit the number of events to the height of the day cell. View a live demo. var calendar = new Calendar(calendarEl, { eventLimit: true, // for all non-TimeGrid views views: { timeGrid: { eventLimit: 6 // adjust to 6 only for timeGridWeek/timeGridDay } }}); Source: Answer by Naomi Burke Documentation: So the question may
be : how to count the events on a given date ?,This is all my way of doing. I am taking left from here. But I am sure, it is not that harder if you try something on the loading: function() of the fullCalendar to limit the number of events according to your wish.,I can use eventRender and return false, but i would like to know how many events are on a given
day, so i can limit the rendering to about 4, then perhaps i would add an event that says " more ... " As of FullCalendar 2.1.0, this feature is included, just use: $('#calendarBlock').fullCalendar({ eventLimit: true [...]}); Source: 3Frq%3D1 Answer by Remy Nielsen i see. you're expecting event limit to work for events that have times and are rendered
within the time slot section? right now eventLimit only works for events within the all-day section as well as for the day tiles within month/basic view.,I'm wondering is eventLimit available only for month view, since I can't see results in other views, for example when I set following config:,Somehow I am also facing same difficulties in my application
using eventLimit in timelineView, agendaWeek etc. However I am able to give you some design(attached) of how I want my UI to be as asked by you in above comment., Have a question about this project? Sign up for a free GitHub account to open an issue and contact its maintainers and the community. eventLimit: 4, views: { basic: { eventLimit: 4 },
agenda: { eventLimit: 4 }, week: { eventLimit: 4 }, day: { eventLimit: 4 } } Source: Answer by Oaklynn Kerr // Code goes here$(document).ready(function() { // page is now ready, initialize the calendar... $('#calendar').fullCalendar({ // put your options and callbacks here header: {left: 'prev,next today',center: 'title',right:
'year,month,basicWeek,basicDay'}, defaultDate: '2014-11-20', timezone: 'local', selectable: true, events: [//At run time, this APEX Repeat will reneder the array elements for the events array { title: "Birmingham Comic Con", start: new Date('2014-11-20T09:00'), end: new Date('2014-11-20T19:00'), id: 1 }, { title: "Birmingham Comic Con", start: new
Date('2014-11-21T09:00'), end: new Date('2014-11-21T19:00'), id: 1 }, { title: "Birmingham Comic Con", start: new Date('2014-11-22T09:00'), end: new Date('2014-11-22T19:00'), id: 1 }] })});function AddEvent(){ $('#calendar').fullCalendar('renderEvent', ourEvent, true);} // Code goes here$(document).ready(function() { // page is now ready,
initialize the calendar... $('#calendar').fullCalendar({ // put your options and callbacks here header: {left: 'prev,next today',center: 'title',right: 'year,month,basicWeek,basicDay'}, defaultDate: '2014-11-20', timezone: 'local', selectable: true, events: [//At run time, this APEX Repeat will reneder the array elements for the events array { title:
"Birmingham Comic Con", start: new Date('2014-11-20T09:00'), end: new Date('2014-11-20T19:00'), id: 1 }, { title: "Birmingham Comic Con", start: new Date('2014-11-21T09:00'), end: new Date('2014-11-21T19:00'), id: 1 }, { title: "Birmingham Comic Con", start: new Date('2014-11-22T09:00'), end: new Date('2014-11-22T19:00'), id: 1 }]
})});function AddEvent(){ $('#calendar').fullCalendar('renderEvent', ourEvent, true);} /* Styles go here */ /* Styles go here */ /*! * FullCalendar v2.4.0 * Docs & License: * (c) 2015 Adam Shaw */(function(factory) {if (typeof define === 'function' && define.amd) {define(['jquery', 'moment'], factory);}else if (typeof exports === 'object') { //
Node/CommonJSmodule.exports = factory(require('jquery'), require('moment'));}else {factory(jQuery, moment);}})(function($, moment) {;;var fc = $.fullCalendar = { version: "2.4.0" };var fcViews = fc.views = {};$.fn.fullCalendar = function(options) {var args = Array.prototype.slice.call(arguments, 1); // for a possible method callvar res = this; //
what this function will return (this jQuery object by default)this.each(function(i, _element) { // loop each DOM element involvedvar element = $(_element);var calendar = element.data('fullCalendar'); // get the existing calendar object (if any)var singleRes; // the returned value of this single method call// a method callif (typeof options === 'string') {if
(calendar && $.isFunction(calendar[options])) {singleRes = calendar[options].apply(calendar, args);if (!i) {res = singleRes; // record the first method call result}if (options === 'destroy') { // for the destroy method, must remove Calendar object dataelement.removeData('fullCalendar');}}}// a new calendar initializationelse if (!calendar) { // don't
initialize twicecalendar = new Calendar(element, options);element.data('fullCalendar', calendar);calendar.render();}});return res;};var complexOptions = [// names of options that are objects whose properties should be combined'header','buttonText','buttonIcons','themeButtonIcons'];// Merges an array of option objects into a single objectfunction
mergeOptions(optionObjs) {return mergeProps(optionObjs, complexOptions);}// Given options specified for the calendar's constructor, massages any legacy options into a non-legacy form.// Converts View-Option-Hashes into the View-Specific-Options format.function massageOverrides(input) {var overrides = { views: input.views || {} }; // the output.
ensure a `views` hashvar subObj;// iterate through all option override properties (except `views`)$.each(input, function(name, val) {if (name != 'views') {// could the value be a legacy View-Option-Hash?if ($.isPlainObject(val) &&!/(time|duration|interval)$/i.test(name) && // exclude duration options. might be given as objects$.inArray(name,
complexOptions) == -1 // complex options aren't allowed to be View-Option-Hashes) {subObj = null;// iterate through the properties of this possible View-Option-Hash value$.each(val, function(subName, subVal) {// is the property targeting a view?if (/^(month|week|day|default|basic(Week|Day)?|agenda(Week|Day)?)$/.test(subName)) {if
(!overrides.views[subName]) { // ensure the view-target entry existsoverrides.views[subName] = {};}overrides.views[subName][name] = subVal; // record the value in the `views` object}else { // a non-View-Option-Hash propertyif (!subObj) {subObj = {};}subObj[subName] = subVal; // accumulate these unrelated values for later}});if (subObj) { //
non-View-Option-Hash properties? transfer them as-isoverrides[name] = subObj;}}else {overrides[name] = val; // transfer normal options as-is}}});return overrides;};;// exportsfc.intersectionToSeg = intersectionToSeg;fc.applyAll = applyAll;fc.debounce = debounce;fc.isInt = isInt;fc.htmlEscape = htmlEscape;fc.cssToStr = cssToStr;fc.proxy =
proxy;fc.capitaliseFirstLetter = capitaliseFirstLetter;/* FullCalendar-specific DOM Utilities--*/// Given the scrollbar widths of some other container, create borders/margins on rowEls in order to match the left// and right space that was offset by the scrollbars. A 1-pixel border
first, then margin beyond that.function compensateScroll(rowEls, scrollbarWidths) {if (scrollbarWidths.left) {rowEls.css({'border-left-width': 1,'margin-left': scrollbarWidths.left - 1});}if (scrollbarWidths.right) {rowEls.css({'border-right-width': 1,'margin-right': scrollbarWidths.right - 1});}}// Undoes compensateScroll and restores all
borders/marginsfunction uncompensateScroll(rowEls) {rowEls.css({'margin-left': '','margin-right': '','border-left-width': '','border-right-width': ''});}// Make the mouse cursor express that an event is not allowed in the current areafunction disableCursor() {$('body').addClass('fc-not-allowed');}// Returns the mouse cursor to its original lookfunction
enableCursor() {$('body').removeClass('fc-not-allowed');}// Given a total available height to fill, have `els` (essentially child rows) expand to accomodate.// By default, all elements that are shorter than the recommended height are expanded uniformly, not considering// any other els that are already too tall. if `shouldRedistribute` is on, it considers
these tall rows and // reduces the available height.function distributeHeight(els, availableHeight, shouldRedistribute) {// *FLOORING NOTE*: we floor in certain places because zoom can give inaccurate floating-point dimensions,// and it is better to be shorter than taller, to avoid creating unnecessary scrollbars.var minOffset1 =
Math.floor(availableHeight / els.length); // for non-last elementvar minOffset2 = Math.floor(availableHeight - minOffset1 * (els.length - 1)); // for last element *FLOORING NOTE*var flexEls = []; // elements that are allowed to expand. array of DOM nodesvar flexOffsets = []; // amount of vertical space it takes upvar flexHeights = []; // actual css
heightvar usedHeight = 0;undistributeHeight(els); // give all elements their natural height// find elements that are below the recommended height (expandable).// important to query for heights in a single first pass (to avoid reflow oscillation).els.each(function(i, el) {var minOffset = i === els.length - 1 ? minOffset2 : minOffset1;var naturalOffset =
$(el).outerHeight(true);if (naturalOffset < minOffset) {flexEls.push(el);flexOffsets.push(naturalOffset);flexHeights.push($(el).height());}else {// this element stretches past recommended height (non-expandable). mark the space as occupied.usedHeight += naturalOffset;}});// readjust the recommended height to only consider the height available to
non-maxed-out rows.if (shouldRedistribute) {availableHeight -= usedHeight;minOffset1 = Math.floor(availableHeight / flexEls.length);minOffset2 = Math.floor(availableHeight - minOffset1 * (flexEls.length - 1)); // *FLOORING NOTE*}// assign heights to all expandable elements$(flexEls).each(function(i, el) {var minOffset = i === flexEls.length - 1 ?
minOffset2 : minOffset1;var naturalOffset = flexOffsets[i];var naturalHeight = flexHeights[i];var newHeight = minOffset - (naturalOffset - naturalHeight); // subtract the margin/paddingif (naturalOffset < minOffset) { // we check this again because redistribution might have changed things$(el).height(newHeight);}});}// Undoes distrubuteHeight,
restoring all els to their natural heightfunction undistributeHeight(els) {els.height('');}// Given `els`, a jQuery set of cells, find the cell with the largest natural width and set the widths of all the// cells to be that width.// PREREQUISITE: if you want a cell to take up width, it needs to have a single inner element w/ display:inlinefunction
matchCellWidths(els) {var maxInnerWidth = 0;els.find('> *').each(function(i, innerEl) {var innerWidth = $(innerEl).outerWidth();if (innerWidth > maxInnerWidth) {maxInnerWidth = innerWidth;}});maxInnerWidth++; // sometimes not accurate of width the text needs to stay on one line. insuranceels.width(maxInnerWidth);return maxInnerWidth;}//
Turns a container element into a scroller if its contents is taller than the allotted height.// Returns true if the element is now a scroller, false otherwise.// NOTE: this method is best because it takes weird zooming dimensions into accountfunction setPotentialScroller(containerEl, height) {containerEl.height(height).addClass('fc-scroller');// are scrollbars
needed?if (containerEl[0].scrollHeight - 1 > containerEl[0].clientHeight) { // !!! -1 because IE is often off-by-one :(return true;}unsetScroller(containerEl); // undoreturn false;}// Takes an element that might have been a scroller, and turns it back into a normal element.function unsetScroller(containerEl) {containerEl.height('').removeClass('fc-
scroller');}/* General DOM Utilities--*/fc.getClientRect = getClientRect;fc.getContentRect = getContentRect;fc.getScrollbarWidths = getScrollbarWidths;// borrowed from getScrollParent(el) {var position = el.css('position'),scrollParent = el.parents().filter(function() {var
parent = $(this);return (/(auto|scroll)/).test(parent.css('overflow') + parent.css('overflow-y') + parent.css('overflow-x'));}).eq(0);return position === 'fixed' || !scrollParent.length ? $(el[0].ownerDocument || document) : scrollParent;}// Queries the outer bounding area of a jQuery element.// Returns a rectangle with absolute coordinates: left, right
(exclusive), top, bottom (exclusive).function getOuterRect(el) {var offset = el.offset();return {left: offset.left,right: offset.left + el.outerWidth(),top: offset.top,bottom: offset.top + el.outerHeight()};}// Queries the area within the margin/border/scrollbars of a jQuery element. Does not go within the padding.// Returns a rectangle with absolute
coordinates: left, right (exclusive), top, bottom (exclusive).// NOTE: should use clientLeft/clientTop, but very unreliable cross-browser.function getClientRect(el) {var offset = el.offset();var scrollbarWidths = getScrollbarWidths(el);var left = offset.left + getCssFloat(el, 'border-left-width') + scrollbarWidths.left;var top = offset.top + getCssFloat(el,
'border-top-width') + scrollbarWidths.top;return {left: left,right: left + el[0].clientWidth, // clientWidth includes padding but NOT scrollbarstop: top,bottom: top + el[0].clientHeight // clientHeight includes padding but NOT scrollbars};}// Queries the area within the margin/border/padding of a jQuery element. Assumed not to have scrollbars.// Returns
a rectangle with absolute coordinates: left, right (exclusive), top, bottom (exclusive).function getContentRect(el) {var offset = el.offset(); // just outside of border, margin not includedvar left = offset.left + getCssFloat(el, 'border-left-width') + getCssFloat(el, 'padding-left');var top = offset.top + getCssFloat(el, 'border-top-width') + getCssFloat(el,
'padding-top');return {left: left,right: left + el.width(),top: top,bottom: top + el.height()};}// Returns the computed left/right/top/bottom scrollbar widths for the given jQuery element.// NOTE: should use clientLeft/clientTop, but very unreliable cross-browser.function getScrollbarWidths(el) {var leftRightWidth = el.innerWidth() - el[0].clientWidth; //
the paddings cancel out, leaving the scrollbarsvar widths = {left: 0,right: 0,top: 0,bottom: el.innerHeight() - el[0].clientHeight // the paddings cancel out, leaving the bottom scrollbar};if (getIsLeftRtlScrollbars() && el.css('direction') == 'rtl') { // is the scrollbar on the left side?widths.left = leftRightWidth;}else {widths.right = leftRightWidth;}return
widths;}// Logic for determining if, when the element is right-to-left, the scrollbar appears on the left sidevar _isLeftRtlScrollbars = null;function getIsLeftRtlScrollbars() { // responsible for caching the computationif (_isLeftRtlScrollbars === null) {_isLeftRtlScrollbars = computeIsLeftRtlScrollbars();}return _isLeftRtlScrollbars;}function
computeIsLeftRtlScrollbars() { // creates an offscreen test element, then removes itvar el = $('').css({position: 'absolute',top: -1000,left: 0,border: 0,padding: 0,overflow: 'scroll',direction: 'rtl'}).appendTo('body');var innerEl = el.children();var res = innerEl.offset().left > el.offset().left; // is the inner div shifted to accommodate a left scrollbar?
el.remove();return res;}// Retrieves a jQuery element's computed CSS value as a floating-point number.// If the queried value is non-numeric (ex: IE can return "medium" for border width), will just return zero.function getCssFloat(el, prop) {return parseFloat(el.css(prop)) || 0;}// Returns a boolean whether this was a left mouse click and no ctrl key
(which means right click on Mac)function isPrimaryMouseButton(ev) {return ev.which == 1 && !ev.ctrlKey;}/* Geometry--*/fc.intersectRects = intersectRects;// Returns a new rectangle that is the intersection of the two rectangles. If they don't intersect, returns
falsefunction intersectRects(rect1, rect2) {var res = {left: Math.max(rect1.left, rect2.left),right: Math.min(rect1.right, rect2.right),top: Math.max(rect1.top, rect2.top),bottom: Math.min(rect1.bottom, rect2.bottom)};if (res.left < res.right && res.top < res.bottom) {return res;}return false;}// Returns a new point that will have been moved to reside
within the given rectanglefunction constrainPoint(point, rect) {return {left: Math.min(Math.max(point.left, rect.left), rect.right),top: Math.min(Math.max(point.top, rect.top), rect.bottom)};}// Returns a point that is the center of the given rectanglefunction getRectCenter(rect) {return {left: (rect.left + rect.right) / 2,top: (rect.top + rect.bottom) /
2};}// Subtracts point2's coordinates from point1's coordinates, returning a deltafunction diffPoints(point1, point2) {return {left: point1.left - point2.left,top: point1.top - point2.top};}/* Object Ordering by Field--*/fc.parseFieldSpecs =
parseFieldSpecs;fc.compareByFieldSpecs = compareByFieldSpecs;fc.compareByFieldSpec = compareByFieldSpec;fc.flexibleCompare = flexibleCompare;function parseFieldSpecs(input) {var specs = [];var tokens = [];var i, token;if (typeof input === 'string') {tokens = input.split(/\s*,\s*/);}else if (typeof input === 'function') {tokens = [input];}else
if ($.isArray(input)) {tokens = input;}for (i = 0; i < tokens.length; i++) {token = tokens[i];if (typeof token === 'string') {specs.push(token.charAt(0) == '-' ?{ field: token.substring(1), order: -1 } :{ field: token, order: 1 });}else if (typeof token === 'function') {specs.push({ func: token });}}return specs;}function compareByFieldSpecs(obj1, obj2,
fieldSpecs) {var i;var cmp;for (i = 0; i < fieldSpecs.length; i++) {cmp = compareByFieldSpec(obj1, obj2, fieldSpecs[i]);if (cmp) {return cmp;}}return 0;}function compareByFieldSpec(obj1, obj2, fieldSpec) {if (fieldSpec.func) {return fieldSpec.func(obj1, obj2);}return flexibleCompare(obj1[fieldSpec.field], obj2[fieldSpec.field]) *(fieldSpec.order ||
1);}function flexibleCompare(a, b) {if (!a && !b) {return 0;}if (b == null) {return -1;}if (a == null) {return 1;}if ($.type(a) === 'string' || $.type(b) === 'string') {return String(a).localeCompare(String(b));}return a - b;}/* FullCalendar-specific Misc Utilities--*/// Creates a
basic segment with the intersection of the two ranges. Returns undefined if no intersection.// Expects all dates to be normalized to the same timezone beforehand.// TODO: move to date section?function intersectionToSeg(subjectRange, constraintRange) {var subjectStart = subjectRange.start;var subjectEnd = subjectRange.end;var constraintStart =
constraintRange.start;var constraintEnd = constraintRange.end;var segStart, segEnd;var isStart, isEnd;if (subjectEnd > constraintStart && subjectStart < constraintEnd) { // in bounds at all?if (subjectStart >= constraintStart) {segStart = subjectStart.clone();isStart = true;}else {segStart = constraintStart.clone();isStart = false;}if (subjectEnd = 1
&& isInt(val)) {break;}}return unit; // will be "milliseconds" if nothing else matches}// Computes the number of units (like "hours") in the given range.// Range can be a {start,end} object, separate start/end args, or a Duration.// Results are based on Moment's .as() and .diff() methods, so results can depend on internal handling// of month-diffing logic
(which tends to vary from version to version).function computeRangeAs(unit, start, end) {if (end != null) { // given start, endreturn end.diff(start, unit, true);}else if (moment.isDuration(start)) { // given durationreturn start.as(unit);}else { // given { start, end } range objectreturn start.end.diff(start.start, unit, true);}}// Intelligently divides a range
(specified by a start/end params) by a durationfunction divideRangeByDuration(start, end, dur) {var months;if (durationHasTime(dur)) {return (end - start) / dur;}months = dur.asMonths();if (Math.abs(months) >= 1 && isInt(months)) {return end.diff(start, 'months', true) / months;}return end.diff(start, 'days', true) / dur.asDays();}// Intelligently
divides one duration by anotherfunction divideDurationByDuration(dur1, dur2) {var months1, months2;if (durationHasTime(dur1) || durationHasTime(dur2)) {return dur1 / dur2;}months1 = dur1.asMonths();months2 = dur2.asMonths();if (Math.abs(months1) >= 1 && isInt(months1) &&Math.abs(months2) >= 1 && isInt(months2)) {return months1 /
months2;}return dur1.asDays() / dur2.asDays();}// Intelligently multiplies a duration by a numberfunction multiplyDuration(dur, n) {var months;if (durationHasTime(dur)) {return moment.duration(dur * n);}months = dur.asMonths();if (Math.abs(months) >= 1 && isInt(months)) {return moment.duration({ months: months * n });}return
moment.duration({ days: dur.asDays() * n });}// Returns a boolean about whether the given duration has any time parts (hours/minutes/seconds/ms)function durationHasTime(dur) {return Boolean(dur.hours() || dur.minutes() || dur.seconds() || dur.milliseconds());}function isNativeDate(input) {return Object.prototype.toString.call(input) === '[object
Date]' || input instanceof Date;}// Returns a boolean about whether the given input is a time string, like "06:40:00" or "06:00"function isTimeString(str) {return /^\d+\:\d+(?:\:\d+\.?(?:\d{3})?)?$/.test(str);}/* Logging and Debug--*/fc.log = function() {var console =
window.console;if (console && console.log) {return console.log.apply(console, arguments);}};fc.warn = function() {var console = window.console;if (console && console.warn) {return console.warn.apply(console, arguments);}else {return fc.log.apply(fc, arguments);}};/* General Utilities--
----------------------*/var hasOwnPropMethod = {}.hasOwnProperty;// Merges an array of objects into a single object.// The second argument allows for an array of property names who's object values will be merged together.function mergeProps(propObjs, complexProps) {var dest = {};var i, name;var complexObjs;var j, val;var props;if (complexProps)
{for (i = 0; i < complexProps.length; i++) {name = complexProps[i];complexObjs = [];// collect the trailing object values, stopping when a non-object is discoveredfor (j = propObjs.length - 1; j >= 0; j--) {val = propObjs[j][name];if (typeof val === 'object') {complexObjs.unshift(val);}else if (val !== undefined) {dest[name] = val; // if there were no
objects, this value will be usedbreak;}}// if the trailing values were objects, use the merged valueif (complexObjs.length) {dest[name] = mergeProps(complexObjs);}}}// copy values into the destination, going from last to firstfor (i = propObjs.length - 1; i >= 0; i--) {props = propObjs[i];for (name in props) {if (!(name in dest)) { // if already assigned by
previous props or complex props, don't reassigndest[name] = props[name];}}}return dest;}// Create an object that has the given prototype. Just like Object.createfunction createObject(proto) {var f = function() {};f.prototype = proto;return new f();}function copyOwnProps(src, dest) {for (var name in src) {if (hasOwnProp(src, name)) {dest[name] =
src[name];}}}// Copies over certain methods with the same names as Object.prototype methods. Overcomes an IE utcOffset(), but don't rely on Moment's internalsthis._ambigTime = false;this._ambigZone = false;return this;};// methods for arbitrarily manipulating timezone offset.// should clear time/zone ambiguity when called.$.each(['zone', // only
in moment-pre-2.9. deprecated afterwards'utcOffset'], function(i, name) {if (oldMomentProto[name]) { // original method exists?// this method implicitly marks a zone (will probably get called upon .utc() and .local())newMomentProto[name] = function(tzo) {if (tzo != null) { // setter// these assignments needs to happen before the original zone method
is called.// I forget why, something to do with a browser crash.this._ambigTime = false;this._ambigZone = false;}return oldMomentProto[name].apply(this, arguments);};}});// Formatting// --newMomentProto.format = function() {if (this._fullCalendar && arguments[0]) { // an enhanced
moment? and a format string provided?return formatDate(this, arguments[0]); // our extended formatting}if (this._ambigTime) {return oldMomentFormat(this, 'YYYY-MM-DD');}if (this._ambigZone) {return oldMomentFormat(this, 'YYYY-MM-DD[T]HH:mm:ss');}return oldMomentProto.format.apply(this, arguments);};newMomentProto.toISOString =
function() {if (this._ambigTime) {return oldMomentFormat(this, 'YYYY-MM-DD');}if (this._ambigZone) {return oldMomentFormat(this, 'YYYY-MM-DD[T]HH:mm:ss');}return oldMomentProto.toISOString.apply(this, arguments);};// Querying// --// Is the moment within the specified range?
`end` is exclusive.// FYI, this method is not a standard Moment method, so always do our enhanced logic.newMomentProto.isWithin = function(start, end) {var a = commonlyAmbiguate([this, start, end]);return a[0] >= a[1] && a[0] < a[2];};// When isSame is called with units, timezone ambiguity is normalized before the comparison happens.// If no
units specified, the two moments must be identically the same, with matching ambig flags.newMomentProto.isSame = function(input, units) {var a;// only do custom logic if this is an enhanced momentif (!this._fullCalendar) {return oldMomentProto.isSame.apply(this, arguments);}if (units) {a = commonlyAmbiguate([this, input], true); // normalize
timezones but don't erase timesreturn oldMomentProto.isSame.call(a[0], a[1], units);}else {input = fc.moment.parseZone(input); // normalize inputreturn oldMomentProto.isSame.call(this, input) &&Boolean(this._ambigTime) === Boolean(input._ambigTime) &&Boolean(this._ambigZone) === Boolean(input._ambigZone);}};// Make these query
methods work with ambiguous moments$.each(['isBefore','isAfter'], function(i, methodName) {newMomentProto[methodName] = function(input, units) {var a;// only do custom logic if this is an enhanced momentif (!this._fullCalendar) {return oldMomentProto[methodName].apply(this, arguments);}a = commonlyAmbiguate([this, input]);return
oldMomentProto[methodName].call(a[0], a[1], units);};});// Misc Internals// --// given an array of moment-like inputs, return a parallel array w/ moments similarly ambiguated.// for example, of one moment has ambig time, but not others, all moments will have their time stripped.// set
`preserveTime` to `true` to keep times, but only normalize zone ambiguity.// returns the original moments if no modifications are necessary.function commonlyAmbiguate(inputs, preserveTime) {var anyAmbigTime = false;var anyAmbigZone = false;var len = inputs.length;var moms = [];var i, mom;// parse inputs into real moments and query their
ambig flagsfor (i = 0; i < len; i++) {mom = inputs[i];if (!moment.isMoment(mom)) {mom = fc.moment.parseZone(mom);}anyAmbigTime = anyAmbigTime || mom._ambigTime;anyAmbigZone = anyAmbigZone || mom._ambigZone;moms.push(mom);}// strip each moment down to lowest common ambiguity// use clones to avoid modifying the original
momentsfor (i = 0; i < len; i++) {mom = moms[i];if (!preserveTime && anyAmbigTime && !mom._ambigTime) {moms[i] = mom.clone().stripTime();}else if (anyAmbigZone && !mom._ambigZone) {moms[i] = mom.clone().stripZone();}}return moms;}// Transfers all the flags related to ambiguous time/zone from the `src` moment to the `dest`
moment// TODO: look into moment.momentProperties for this.function transferAmbigs(src, dest) {if (src._ambigTime) {dest._ambigTime = true;}else if (dest._ambigTime) {dest._ambigTime = false;}if (src._ambigZone) {dest._ambigZone = true;}else if (dest._ambigZone) {dest._ambigZone = false;}}// Sets the year/month/date/etc values of the
moment from the given array.// Inefficient because it calls each individual setter.function setMomentValues(mom, a) {mom.year(a[0] || 0).month(a[1] || 0).date(a[2] || 0).hours(a[3] || 0).minutes(a[4] || 0).seconds(a[5] || 0).milliseconds(a[6] || 0);}// Can we set the moment's internal date directly?allowValueOptimization = '_d' in moment() &&
'updateOffset' in moment;// Utility function. Accepts a moment and an array of the UTC year/month/date/etc values to set.// Assumes the given moment is already in UTC mode.setUTCValues = allowValueOptimization ? function(mom, a) {// simlate what moment's accessors domom._d.setTime(Date.UTC.apply(Date, a));moment.updateOffset(mom,
false); // keepTime=false} : setMomentValues;// Utility function. Accepts a moment and an array of the local year/month/date/etc values to set.// Assumes the given moment is already in local mode.setLocalValues = allowValueOptimization ? function(mom, a) {// simlate what moment's accessors domom._d.setTime(+new Date(// FYI, there is now way
to apply an array of args to a constructora[0] || 0,a[1] || 0,a[2] || 0,a[3] || 0,a[4] || 0,a[5] || 0,a[6] || 0));moment.updateOffset(mom, false); // keepTime=false} : setMomentValues;;;// Single Date Formatting// --// call this if you want Moment's original format method to be usedfunction
oldMomentFormat(mom, formatStr) {return oldMomentProto.format.call(mom, formatStr); // oldMomentProto defined in moment-ext.js}// Formats `date` with a Moment formatting string, but allow our non-zero areas and// additional token.function formatDate(date, formatStr) {return formatDateWithChunks(date,
getFormatStringChunks(formatStr));}function formatDateWithChunks(date, chunks) {var s = '';var i;for (i=0; i "MMMM D YYYY"formatStr = localeData.longDateFormat(formatStr) || formatStr;// BTW, this is not important for `formatDate` because it is impossible to put custom tokens// or non-zero areas in Moment's localized format strings.separator
= separator || ' - ';return formatRangeWithChunks(date1,date2,getFormatStringChunks(formatStr),separator,isRTL);}fc.formatRange = formatRange; // exposefunction formatRangeWithChunks(date1, date2, chunks, separator, isRTL) {var chunkStr; // the rendering of the chunkvar leftI;var leftStr = '';var rightI;var rightStr = '';var middleI;var
middleStr1 = '';var middleStr2 = '';var middleStr = '';// Start at the leftmost side of the formatting string and continue until you hit a token// that is not the same between dates.for (leftI=0; leftIleftI; rightI--) {chunkStr = formatSimilarChunk(date1, date2, chunks[rightI]);if (chunkStr === false) {break;}rightStr = chunkStr + rightStr;}// The area in
the middle is different for both of the dates.// Collect them distinctly so we can jam them together later.for (middleI=leftI; middleI= coords.top && y < coords.bottom) {hitRow = i;break;}}for (i = 0; i < colCnt; i++) {coords = colCoords[i];if (x >= coords.left && x < coords.right) {hitCol = i;break;}}if (hitRow !== null && hitCol !== null) {cell =
this.grid.getCell(hitRow, hitCol); // expected to return a fresh object we can modifycell.grid = this.grid; // for CellDragListener's isCellsEqual. dragging between grids// make the coordinates available on the cell object$.extend(cell, rowCoords[hitRow], colCoords[hitCol]);return cell;}}return null;},// If there is a containerEl, compute the bounds into
min/max valuescomputeBounds: function() {this.bounds = this.containerEl ?getClientRect(this.containerEl) : // area within scrollbarsnull;},// Determines if the given coordinates are in bounds. If no `containerEl`, always trueinBounds: function(x, y) {var bounds = this.bounds;if (bounds) {return x >= bounds.left && x < bounds.right && y >=
bounds.top && y < bounds.bottom;}return true;}});/* Coordinate map that is a combination of multiple other coordinate maps--*/var ComboCoordMap = Class.extend({coordMaps: null, // an array of CoordMapsconstructor: function(coordMaps) {this.coordMaps =
coordMaps;},// Builds all coordMapsbuild: function() {var coordMaps = this.coordMaps;var i;for (i = 0; i < coordMaps.length; i++) {coordMaps[i].build();}},// Queries all coordMaps for the cell underneath the given coordinates, returning the first resultgetCell: function(x, y) {var coordMaps = this.coordMaps;var cell = null;var i;for (i = 0; i <
coordMaps.length && !cell; i++) {cell = coordMaps[i].getCell(x, y);}return cell;},// Clears all coordMapsclear: function() {var coordMaps = this.coordMaps;var i;for (i = 0; i < coordMaps.length; i++) {coordMaps[i].clear();}}});;;/* Tracks a drag's mouse movement, firing various handlers--
----------------------*/var DragListener = fc.DragListener = Class.extend({options: null,isListening: false,isDragging: false,// coordinates of the initial mousedownoriginX: null,originY: null,// handler attached to the document, bound to the DragListener's `this`mousemoveProxy: null,mouseupProxy: null,// for IE8 bug-fighting behavior, for nowsubjectEl:
null, // the element being draged. optionalsubjectHref: null,scrollEl: null,scrollBounds: null, // { top, bottom, left, right }scrollTopVel: null, // pixels per secondscrollLeftVel: null, // pixels per secondscrollIntervalId: null, // ID of setTimeout for scrolling animation loopscrollHandlerProxy: null, // this-scoped function for handling when scrollEl is
scrolledscrollSensitivity: 30, // pixels from edge for scrolling to startscrollSpeed: 200, // pixels per second, at maximum speedscrollIntervalMs: 50, // millisecond wait between scroll incrementconstructor: function(options) {options = options || {};this.options = options;this.subjectEl = options.subjectEl;},// Call this when the user does a mousedown.
Will probably lead to startListeningmousedown: function(ev) {if (isPrimaryMouseButton(ev)) {ev.preventDefault(); // prevents native selection in most browsersthis.startListening(ev);// start the drag immediately if there is no minimum distance for a drag startif (!this.options.distance) {this.startDrag(ev);}}},// Call this to start tracking mouse
movementsstartListening: function(ev) {var scrollParent;if (!this.isListening) {// grab scroll container and attach handlerif (ev && this.options.scroll) {scrollParent = getScrollParent($(ev.target));if (!scrollParent.is(window) && !scrollParent.is(document)) {this.scrollEl = scrollParent;// scope to `this`, and use `debounce` to make sure rapid calls don't
happenthis.scrollHandlerProxy = debounce(proxy(this, 'scrollHandler'), 100);this.scrollEl.on('scroll', this.scrollHandlerProxy);}}$(document).on('mousemove', this.mousemoveProxy = proxy(this, 'mousemove')).on('mouseup', this.mouseupProxy = proxy(this, 'mouseup')).on('selectstart', this.preventDefault); // prevents native selection in IE=
minDistance * minDistance) { // use pythagorean theoremthis.startDrag(ev);}}if (this.isDragging) {this.drag(dx, dy, ev); // report a drag, even if this mousemove initiated the drag}},// Call this to initiate a legitimate drag.// This function is called internally from this class, but can also be called explicitly from outsidestartDrag: function(ev) {if
(!this.isListening) { // startDrag must have manually initiatedthis.startListening();}if (!this.isDragging) {this.isDragging = true;this.dragStart(ev);}},// Called when the actual drag has started (went beyond minDistance)dragStart: function(ev) {var subjectEl = this.subjectEl;this.trigger('dragStart', ev);// remove a mousedown'd 's href so it is not visited
(IE8 bug)if ((this.subjectHref = subjectEl ? subjectEl.attr('href') : null)) {subjectEl.removeAttr('href');}},// Called while the mouse is being moved and when we know a legitimate drag is taking placedrag: function(dx, dy, ev) {this.trigger('drag', dx, dy, ev);this.updateScroll(ev); // will possibly cause scrolling},// Called when the user does a
mouseupmouseup: function(ev) {this.stopListening(ev);},// Called when the drag is over. Will not cause listening to stop however.// A concluding 'cellOut' event will NOT be triggered.stopDrag: function(ev) {if (this.isDragging) {this.stopScrolling();this.dragStop(ev);this.isDragging = false;}},// Called when dragging has been stoppeddragStop:
function(ev) {var _this = this;this.trigger('dragStop', ev);// restore a mousedown'd 's href (for IE8 bug)setTimeout(function() { // must be outside of the click's executionif (_this.subjectHref) {_this.subjectEl.attr('href', _this.subjectHref);}}, 0);},// Call this to stop listening to the user's mouse eventsstopListening: function(ev) {this.stopDrag(ev); // if
there's a current drag, kill itif (this.isListening) {// remove the scroll handler if there is a scrollElif (this.scrollEl) {this.scrollEl.off('scroll', this.scrollHandlerProxy);this.scrollHandlerProxy = null;}$(document).off('mousemove', this.mousemoveProxy).off('mouseup', this.mouseupProxy).off('selectstart', this.preventDefault);this.mousemoveProxy =
null;this.mouseupProxy = null;this.isListening = false;this.listenStop(ev);}},// Called when drag listening has stoppedlistenStop: function(ev) {this.trigger('listenStop', ev);},// Triggers a callback. Calls a function in the option hash of the same name.// Arguments beyond the first `name` are forwarded on.trigger: function(name) {if (this.options[name])
{this.options[name].apply(this, Array.prototype.slice.call(arguments, 1));}},// Stops a given mouse event from doing it's native browser action. In our case, text selection.preventDefault: function(ev) {ev.preventDefault();},/* Scrolling--*/// Computes and stores the bounding
rectangle of scrollElcomputeScrollBounds: function() {var el = this.scrollEl;this.scrollBounds = el ? getOuterRect(el) : null;// TODO: use getClientRect in future. but prevents auto scrolling when on top of scrollbars},// Called when the dragging is in progress and scrolling should be updatedupdateScroll: function(ev) {var sensitivity =
this.scrollSensitivity;var bounds = this.scrollBounds;var topCloseness, bottomCloseness;var leftCloseness, rightCloseness;var topVel = 0;var leftVel = 0;if (bounds) { // only scroll if scrollEl exists// compute closeness to edges. valid range is from 0.0 - 1.0topCloseness = (sensitivity - (ev.pageY - bounds.top)) / sensitivity;bottomCloseness = (sensitivity -
(bounds.bottom - ev.pageY)) / sensitivity;leftCloseness = (sensitivity - (ev.pageX - bounds.left)) / sensitivity;rightCloseness = (sensitivity - (bounds.right - ev.pageX)) / sensitivity;// translate vertical closeness into velocity.// mouse must be completely in bounds for velocity to happen.if (topCloseness >= 0 && topCloseness = 0 && bottomCloseness = 0
&& leftCloseness = 0 && rightCloseness *', function(ev) {var seg = $(this).data('fc-seg'); // grab segment data. put there by View::renderEvents// only call the handlers if there is not a drag/resize in progressif (seg && !_this.isDraggingSeg && !_this.isResizingSeg) {return func.call(this, seg, ev); // `this` will be the event element}});});},// Updates
internal state and triggers handlers for when an event element is moused overtriggerSegMouseover: function(seg, ev) {if (!this.mousedOverSeg) {this.mousedOverSeg = seg;this.view.trigger('eventMouseover', seg.el[0], seg.event, ev);}},// Updates internal state and triggers handlers for when an event element is moused out.// Can be given no
arguments, in which case it will mouseout the segment that was previously moused over.triggerSegMouseout: function(seg, ev) {ev = ev || {}; // if given no args, make a mock mouse eventif (this.mousedOverSeg) {seg = seg || this.mousedOverSeg; // if given no args, use the currently moused-over segmentthis.mousedOverSeg =
null;this.view.trigger('eventMouseout', seg.el[0], seg.event, ev);}},/* Event Dragging--*/// Called when the user does a mousedown on an event, which might lead to dragging.// Generic enough to work with any type of Grid.segDragMousedown: function(seg, ev) {var _this =
this;var view = this.view;var calendar = view.calendar;var el = seg.el;var event = seg.event;var dropLocation;// A clone of the original element that will move with the mousevar mouseFollower = new MouseFollower(seg.el, {parentEl: view.el,opacity: view.opt('dragOpacity'),revertDuration: view.opt('dragRevertDuration'),zIndex: 2 // one above the .fc-
view});// Tracks mouse movement over the *view's* coordinate map. Allows dragging and dropping between subcomponents// of the view.var dragListener = new CellDragListener(view.coordMap, {distance: 5,scroll: view.opt('dragScroll'),subjectEl: el,subjectCenter: true,listenStart: function(ev) {mouseFollower.hide(); // don't show until we know this
is a real dragmouseFollower.start(ev);},dragStart: function(ev) {_this.triggerSegMouseout(seg, ev); // ensure a mouseout on the manipulated event has been reported_this.segDragStart(seg, ev);view.hideEvent(event); // hide all event segments. our mouseFollower will take over},cellOver: function(cell, isOrig, origCell) {// starting cell could be forced
(DayGrid.limit)if (seg.cell) {origCell = seg.cell;}dropLocation = _this.computeEventDrop(origCell, cell, event);if (dropLocation && !calendar.isEventRangeAllowed(dropLocation, event)) {disableCursor();dropLocation = null;}// if a valid drop location, have the subclass render a visual indicationif (dropLocation && view.renderDrag(dropLocation, seg))
{mouseFollower.hide(); // if the subclass is already using a mock event "helper", hide our own}else {mouseFollower.show(); // otherwise, have the helper follow the mouse (no snapping)}if (isOrig) {dropLocation = null; // needs to have moved cells to be a valid drop}},cellOut: function() { // called before mouse moves to a different cell OR moved out
of all cellsview.unrenderDrag(); // unrender whatever was done in renderDragmouseFollower.show(); // show in case we are moving out of all cellsdropLocation = null;},cellDone: function() { // Called after a cellOut OR before a dragStopenableCursor();},dragStop: function(ev) {// do revert animation if hasn't changed. calls a callback when finished
(whether animation or not)mouseFollower.stop(!dropLocation, function() {view.unrenderDrag();view.showEvent(event);_this.segDragStop(seg, ev);if (dropLocation) {view.reportEventDrop(event, dropLocation, this.largeUnit, el, ev);}});},listenStop: function() {mouseFollower.stop(); // put in listenStop in case there was a mousedown but the drag
never started}});dragListener.mousedown(ev); // start listening, which will eventually lead to a dragStart},// Called before event segment dragging startssegDragStart: function(seg, ev) {this.isDraggingSeg = true;this.view.trigger('eventDragStart', seg.el[0], seg.event, ev, {}); // last argument is jqui dummy},// Called after event segment dragging
stopssegDragStop: function(seg, ev) {this.isDraggingSeg = false;this.view.trigger('eventDragStop', seg.el[0], seg.event, ev, {}); // last argument is jqui dummy},// Given the cell an event drag began, and the cell event was dropped, calculates the new start/end/allDay// values for the event. Subclasses may override and set additional properties to be
used by renderDrag.// A falsy returned value indicates an invalid drop.computeEventDrop: function(startCell, endCell, event) {var calendar = this.view.calendar;var dragStart = startCell.start;var dragEnd = endCell.start;var delta;var dropLocation;if (dragStart.hasTime() === dragEnd.hasTime()) {delta = this.diffDates(dragEnd, dragStart);// if an all-
day event was in a timed area and it was dragged to a different time,// guarantee an end and adjust start/end to have timesif (event.allDay && durationHasTime(delta)) {dropLocation = {start: event.start.clone(),end: calendar.getEventEnd(event), // will be an ambig dayallDay: false // for
normalizeEventRangeTimes};calendar.normalizeEventRangeTimes(dropLocation);}// othewise, work off existing valueselse {dropLocation = {start: event.start.clone(),end: event.end ? event.end.clone() : null,allDay: event.allDay // keep it the same};}dropLocation.start.add(delta);if (dropLocation.end) {dropLocation.end.add(delta);}}else {// if
switching from day timed, start should be reset to the dropped date, and the end cleareddropLocation = {start: dragEnd.clone(),end: null, // end should be clearedallDay: !dragEnd.hasTime()};}return dropLocation;},// Utility for apply dragOpacity to a jQuery setapplyDragOpacity: function(els) {var opacity = this.view.opt('dragOpacity');if (opacity !=
null) {els.each(function(i, node) {// Don't use jQuery (will set an IE filter), do it the old fashioned way.// In IE8, a helper element will disappears if there's a filter.node.style.opacity = opacity;});}},/* External Element Dragging--*/// Called when a jQuery UI drag is initiated
anywhere in the DOMexternalDragStart: function(ev, ui) {var view = this.view;var el;var accept;if (view.opt('droppable')) { // only listen if this setting is onel = $((ui ? ui.item : null) || ev.target);// Test that the dragged element passes the dropAccept selector or filter function.// FYI, the default is "*" (matches all)accept = view.opt('dropAccept');if
($.isFunction(accept) ? accept.call(el[0], el) : el.is(accept)) {if (!this.isDraggingExternal) { // prevent double-listening if fired twicethis.listenToExternalDrag(el, ev, ui);}}}},// Called when a jQuery UI drag starts and it needs to be monitored for cell droppinglistenToExternalDrag: function(el, ev, ui) {var _this = this;var meta = getDraggedElMeta(el); //
extra data about event drop, including possible event to createvar dragListener;var dropLocation; // a null value signals an unsuccessful drag// listener that tracks mouse movement over date-associated pixel regionsdragListener = new CellDragListener(this.coordMap, {listenStart: function() {_this.isDraggingExternal = true;},cellOver: function(cell)
{dropLocation = _this.computeExternalDrop(cell, meta);if (dropLocation) {_this.renderDrag(dropLocation); // called without a seg parameter}else { // invalid drop celldisableCursor();}},cellOut: function() {dropLocation = null; // signal unsuccessful_this.unrenderDrag();enableCursor();},dragStop: function() {_this.unrenderDrag();enableCursor();if
(dropLocation) { // element was dropped on a valid date/time cell_this.view.reportExternalDrop(meta, dropLocation, el, ev, ui);}},listenStop: function() {_this.isDraggingExternal = false;}});dragListener.startDrag(ev); // start listening immediately},// Given a cell to be dropped upon, and misc data associated with the jqui drag (guaranteed to be a
plain object),// returns start/end dates for the event that would result from the hypothetical drop. end might be null.// Returning a null value signals an invalid drop cell.computeExternalDrop: function(cell, meta) {var dropLocation = {start: cell.start.clone(),end: null};// if dropped on an all-day cell, and element's metadata specified a time, set itif
(meta.startTime && !dropLocation.start.hasTime()) {dropLocation.start.time(meta.startTime);}if (meta.duration) {dropLocation.end = dropLocation.start.clone().add(meta.duration);}if (!this.view.calendar.isExternalDropRangeAllowed(dropLocation, meta.eventProps)) {return null;}return dropLocation;},/* Drag Rendering (for both events and
external elements)--*/// Renders a visual indication of an event or external element being dragged.// `dropLocation` contains hypothetical start/end/allDay values the event would have if dropped. end can be null.// `seg` is the internal segment object that is being dragged. If
dragging an external element, `seg` is null.// A truthy returned value indicates this method has rendered a helper element.renderDrag: function(dropLocation, seg) {// subclasses must implement},// Unrenders a visual indication of an event or external element being draggedunrenderDrag: function() {// subclasses must implement},/* Resizing------------
--*/// Called when the user does a mousedown on an event's resizer, which might lead to resizing.// Generic enough to work with any type of Grid.segResizeMousedown: function(seg, ev, isStart) {var _this = this;var view = this.view;var calendar = view.calendar;var el = seg.el;var event
= seg.event;var eventEnd = calendar.getEventEnd(event);var dragListener;var resizeLocation; // falsy if invalid resize// Tracks mouse movement over the *grid's* coordinate mapdragListener = new CellDragListener(this.coordMap, {distance: 5,scroll: view.opt('dragScroll'),subjectEl: el,dragStart: function(ev) {_this.triggerSegMouseout(seg, ev); //
ensure a mouseout on the manipulated event has been reported_this.segResizeStart(seg, ev);},cellOver: function(cell, isOrig, origCell) {resizeLocation = isStart ?_this.computeEventStartResize(origCell, cell, event) :_this.computeEventEndResize(origCell, cell, event);if (resizeLocation) {if (!calendar.isEventRangeAllowed(resizeLocation, event))
{disableCursor();resizeLocation = null;}// no change? (TODO: how does this work with timezones?)else if (resizeLocation.start.isSame(event.start) && resizeLocation.end.isSame(eventEnd)) {resizeLocation = null;}}if (resizeLocation) {view.hideEvent(event);_this.renderEventResize(resizeLocation, seg);}},cellOut: function() { // called before mouse
moves to a different cell OR moved out of all cellsresizeLocation = null;},cellDone: function() { // resets the rendering to show the original event_this.unrenderEventResize();view.showEvent(event);enableCursor();},dragStop: function(ev) {_this.segResizeStop(seg, ev);if (resizeLocation) { // valid date to resize to?view.reportEventResize(event,
resizeLocation, this.largeUnit, el, ev);}}});dragListener.mousedown(ev); // start listening, which will eventually lead to a dragStart},// Called before event segment resizing startssegResizeStart: function(seg, ev) {this.isResizingSeg = true;this.view.trigger('eventResizeStart', seg.el[0], seg.event, ev, {}); // last argument is jqui dummy},// Called after
event segment resizing stopssegResizeStop: function(seg, ev) {this.isResizingSeg = false;this.view.trigger('eventResizeStop', seg.el[0], seg.event, ev, {}); // last argument is jqui dummy},// Returns new date-information for an event segment being resized from its startcomputeEventStartResize: function(startCell, endCell, event) {return
this.computeEventResize('start', startCell, endCell, event);},// Returns new date-information for an event segment being resized from its endcomputeEventEndResize: function(startCell, endCell, event) {return this.computeEventResize('end', startCell, endCell, event);},// Returns new date-information for an event segment being resized from its start
OR end// `type` is either 'start' or 'end'computeEventResize: function(type, startCell, endCell, event) {var calendar = this.view.calendar;var delta = this.diffDates(endCell[type], startCell[type]);var range;var defaultDuration;// build original values to work from, guaranteeing a start and endrange = {start: event.start.clone(),end:
calendar.getEventEnd(event),allDay: event.allDay};// if an all-day event was in a timed area and was resized to a time, adjust start/end to have timesif (range.allDay && durationHasTime(delta)) {range.allDay = false;calendar.normalizeEventRangeTimes(range);}range[type].add(delta); // apply delta to start or end// if the event was compressed too
small, find a new reasonable duration for itif (!range.start.isBefore(range.end)) {defaultDuration = event.allDay ?calendar.defaultAllDayEventDuration :calendar.defaultTimedEventDuration;// between the cell's duration and the event's default duration, use the smaller of the two.// example: if year-length slots, and compressed to one slot, we don't
want the event to be a year longif (this.cellDuration && this.cellDuration < defaultDuration) {defaultDuration = this.cellDuration;}if (type == 'start') { // resizing the start?range.start = range.end.clone().subtract(defaultDuration);}else { // resizing the end?range.end = range.start.clone().add(defaultDuration);}}return range;},// Renders a visual
indication of an event being resized.// `range` has the updated dates of the event. `seg` is the original segment object involved in the drag.renderEventResize: function(range, seg) {// subclasses must implement},// Unrenders a visual indication of an event being resized.unrenderEventResize: function() {// subclasses must implement},/* Rendering
Utils--*/// Compute the text that should be displayed on an event's element.// `range` can be the Event object itself, or something range-like, with at least a `start`.// If event times are disabled, or the event has no time, will return a blank string.// If not specified, formatStr will
default to the eventTimeFormat setting,// and displayEnd will default to the displayEventEnd setting.getEventTimeText: function(range, formatStr, displayEnd) {if (formatStr == null) {formatStr = this.eventTimeFormat;}if (displayEnd == null) {displayEnd = this.displayEventEnd;}if (this.displayEventTime && range.start.hasTime()) {if (displayEnd
&& range.end) {return this.view.formatRange(range, formatStr);}else {return range.start.format(formatStr);}}return '';},// Generic utility for generating the HTML classNames for an event segment's elementgetSegClasses: function(seg, isDraggable, isResizable) {var event = seg.event;var classes = ['fc-event',seg.isStart ? 'fc-start' : 'fc-not-
start',seg.isEnd ? 'fc-end' : 'fc-not-end'].concat(event.className,event.source ? event.source.className : []);if (isDraggable) {classes.push('fc-draggable');}if (isResizable) {classes.push('fc-resizable');}return classes;},// Utility for generating event skin-related CSS propertiesgetEventSkinCss: function(event) {var view = this.view;var source =
event.source || {};var eventColor = event.color;var sourceColor = source.color;var optionColor = view.opt('eventColor');return {'background-color':event.backgroundColor ||eventColor ||source.backgroundColor ||sourceColor ||view.opt('eventBackgroundColor') ||optionColor,'border-color':event.borderColor ||eventColor ||source.borderColor
||sourceColor ||view.opt('eventBorderColor') ||optionColor,color:event.textColor ||source.textColor ||view.opt('eventTextColor')};},/* Converting events -> ranges -> segs--*/// Converts an array of event objects into an array of event segment objects.// A custom
`rangeToSegsFunc` may be given for arbitrarily slicing up events.// Doesn't guarantee an order for the resulting array.eventsToSegs: function(events, rangeToSegsFunc) {var eventRanges = this.eventsToRanges(events);var segs = [];var i;for (i = 0; i < eventRanges.length; i++) {segs.push.apply(segs,this.eventRangeToSegs(eventRanges[i],
rangeToSegsFunc));}return segs;},// Converts an array of events into an array of "range" objects.// A "range" object is a plain object with start/end properties denoting the time it covers. Also an event property.// For "normal" events, this will be identical to the event's start/end, but for "inverse-background" events,// will create an array of ranges that
span the time *not* covered by the given event.// Doesn't guarantee an order for the resulting array.eventsToRanges: function(events) {var _this = this;var eventsById = groupEventsById(events);var ranges = [];// group by ID so that related inverse-background events can be rendered together$.each(eventsById, function(id, eventGroup) {if
(eventGroup.length) {ranges.push.apply(ranges,isInverseBgEvent(eventGroup[0]) ?_this.eventsToInverseRanges(eventGroup) :_this.eventsToNormalRanges(eventGroup));}});return ranges;},// Converts an array of "normal" events (not inverted rendering) into a parallel array of rangeseventsToNormalRanges: function(events) {var calendar =
this.view.calendar;var ranges = [];var i, event;var eventStart, eventEnd;for (i = 0; i < events.length; i++) {event = events[i];// make copies and normalize by stripping timezoneeventStart = event.start.clone().stripZone();eventEnd = calendar.getEventEnd(event).stripZone();ranges.push({event: event,start: eventStart,end: eventEnd,eventStartMS:
+eventStart,eventDurationMS: eventEnd - eventStart});}return ranges;},// Converts an array of events, with inverse-background rendering, into an array of range objects.// The range objects will cover all the time NOT covered by the events.eventsToInverseRanges: function(events) {var view = this.view;var viewStart = view.start.clone().stripZone();
// normalize timezonevar viewEnd = view.end.clone().stripZone(); // normalize timezonevar normalRanges = this.eventsToNormalRanges(events); // will give us normalized dates we can use w/o copiesvar inverseRanges = [];var event0 = events[0]; // assign this to each range's `.event`var start = viewStart; // the end of the previous range. the start of
the new rangevar i, normalRange;// ranges need to be in order. required for our date-walking algorithmnormalRanges.sort(compareNormalRanges);for (i = 0; i < normalRanges.length; i++) {normalRange = normalRanges[i];// add the span of time before the event (if there is any)if (normalRange.start > start) { // compare millisecond time (skip any
ambig logic)inverseRanges.push({event: event0,start: start,end: normalRange.start});}start = normalRange.end;}// add the span of time after the last event (if there is any)if (start < viewEnd) { // compare millisecond time (skip any ambig logic)inverseRanges.push({event: event0,start: start,end: viewEnd});}return inverseRanges;},// Slices the given
event range into one or more segment objects.// A `rangeToSegsFunc` custom slicing function can be given.eventRangeToSegs: function(eventRange, rangeToSegsFunc) {var segs;var i, seg;eventRange = this.view.calendar.ensureVisibleEventRange(eventRange);if (rangeToSegsFunc) {segs = rangeToSegsFunc(eventRange);}else {segs =
this.rangeToSegs(eventRange); // defined by the subclass}for (i = 0; i < segs.length; i++) {seg = segs[i];seg.event = eventRange.event;seg.eventStartMS = eventRange.eventStartMS;seg.eventDurationMS = eventRange.eventDurationMS;}return segs;},sortSegs: function(segs) {segs.sort(proxy(this, 'compareSegs'));},// A cmp function for
determining which segments should take visual priority// DOES NOT WORK ON INVERTED BACKGROUND EVENTS because they have no eventStartMS/eventDurationMScompareSegs: function(seg1, seg2) {return seg1.eventStartMS - seg2.eventStartMS || // earlier events go firstseg2.eventDurationMS - seg1.eventDurationMS || // tie? longer events
go firstseg2.event.allDay - seg1.event.allDay || // tie? put all-day events first (booleans cast to 0/1)compareByFieldSpecs(seg1.event, seg2.event, this.view.eventOrderSpecs);}});/* Utilities--*/function isBgEvent(event) { // returns true if background OR inverse-backgroundvar
rendering = getEventRendering(event);return rendering === 'background' || rendering === 'inverse-background';}function isInverseBgEvent(event) {return getEventRendering(event) === 'inverse-background';}function getEventRendering(event) {return firstDefined((event.source || {}).rendering, event.rendering);}function
groupEventsById(events) {var eventsById = {};var i, event;for (i = 0; i < events.length; i++) {event = events[i];(eventsById[event._id] || (eventsById[event._id] = [])).push(event);}return eventsById;}// A cmp function for determining which non-inverted "ranges" (see above) happen earlierfunction compareNormalRanges(range1, range2) {return
range1.eventStartMS - range2.eventStartMS; // earlier ranges go first}/* External-Dragging-Element Data--*/// Require all HTML5 data-* attributes used by FullCalendar to have this prefix.// A value of '' will query attributes like data-event. A value of 'fc' will query attributes
like data-fc-event.fc.dataAttrPrefix = '';// Given a jQuery element that might represent a dragged FullCalendar event, returns an intermediate data structure// to be used for Event Object creation.// A defined `.eventProps`, even when empty, indicates that an event should be created.function getDraggedElMeta(el) {var prefix = fc.dataAttrPrefix;var
eventProps; // properties for creating the event, not related to date/timevar startTime; // a Durationvar duration;var stick;if (prefix) { prefix += '-'; }eventProps = el.data(prefix + 'event') || null;if (eventProps) {if (typeof eventProps === 'object') {eventProps = $.extend({}, eventProps); // make a copy}else { // something like 1 or true. still signal event
creationeventProps = {};}// pluck special-cased date/time propertiesstartTime = eventProps.start;if (startTime == null) { startTime = eventProps.time; } // accept 'time' as wellduration = eventProps.duration;stick = eventProps.stick;delete eventProps.start;delete eventProps.time;delete eventProps.duration;delete eventProps.stick;}// fallback to
standalone attribute values for each of the date/time propertiesif (startTime == null) { startTime = el.data(prefix + 'start'); }if (startTime == null) { startTime = el.data(prefix + 'time'); } // accept 'time' as wellif (duration == null) { duration = el.data(prefix + 'duration'); }if (stick == null) { stick = el.data(prefix + 'stick'); }// massage into correct
data typesstartTime = startTime != null ? moment.duration(startTime) : null;duration = duration != null ? moment.duration(duration) : null;stick = Boolean(stick);return { eventProps: eventProps, startTime: startTime, duration: duration, stick: stick };};;/* A component that renders a grid of whole-days that runs horizontally. There can be multiple
rows, one per week.--*/var DayGrid = Grid.extend({numbersVisible: false, // should render a row for day/week numbers? set by outside view. TODO: make internalbottomCoordPadding: 0, // hack for extending the hit area for the last row of the coordinate gridbreakOnWeeks:
null, // should create a new row for each week? set by outside viewcellDates: null, // flat chronological array of each cell's datesdayToCellOffsets: null, // maps days offsets from grid's start date, to cell offsetsrowEls: null, // set of fake row elementsdayEls: null, // set of whole-day elements comprising the row's backgroundhelperEls: null, // set of cell
skeleton elements for rendering the mock event "helper"constructor: function() {Grid.apply(this, arguments);this.cellDuration = moment.duration(1, 'day'); // for Grid system},// Renders the rows and columns into the component's `this.el`, which should already be assigned.// isRigid determins whether the individual rows should ignore the contents
and be a constant height.// Relies on the view's colCnt and rowCnt. In the future, this component should probably be self-sufficient.renderDates: function(isRigid) {var view = this.view;var rowCnt = this.rowCnt;var colCnt = this.colCnt;var cellCnt = rowCnt * colCnt;var html = '';var row;var i, cell;for (row = 0; row < rowCnt; row++) {html +=
this.dayRowHtml(row, isRigid);}this.el.html(html);this.rowEls = this.el.find('.fc-row');this.dayEls = this.el.find('.fc-day');// trigger dayRender with each cell's elementfor (i = 0; i < cellCnt; i++) {cell = this.getCell(i);view.trigger('dayRender', null, cell.start, this.dayEls.eq(i));}},unrenderDates: function()

```
{this.removeSegPopover();},renderBusinessHours: function() {var events = this.view.calendar.getBusinessHoursEvents(true); // wholeDay=truevar segs = this.eventsToSegs(events);this.renderFill('businessHours', segs, 'bgevent');},// Generates the HTML for a single row. `row` is the row number.dayRowHtml: function(row, isRigid) {var view =
this.view;var classes = [ 'fc-row', 'fc-week', view.widgetContentClass ];if (isRigid) {classes.push('fc-rigid');}return '' +'' +'' +'' +this.rowHtml('day', row) + // leverages RowRenderer. calls dayCellHtml()'' +'' +'' +'' +(this.numbersVisible ?'' +this.rowHtml('number', row) + // leverages RowRenderer. View will define render method'' :'') +'' +'' +'';},//
Renders the HTML for a whole-day cell. Will eventually end up in the day-row's background.// We go through a 'day' row type instead of just doing a 'bg' row type so that the View can do custom rendering// specifically for whole-day rows, whereas a 'bg' might also be used for other purposes (TimeGrid bg for example).dayCellHtml: function(cell)
{return this.bgCellHtml(cell);},/* Options------------------------------------------------------------------------------------------------------------------*/// Computes a default column header formatting string if `colFormat` is not explicitly definedcomputeColHeadFormat: function() {if (this.rowCnt > 1) { // more than one week row. day numbers will be in each cellreturn 'ddd'; //
"Sat"}else if (this.colCnt > 1) { // multiple days, so full single date string WON'T be in title textreturn this.view.opt('dayOfMonthFormat'); // "Sat 12/10"}else { // single day, so full single date string will probably be in title textreturn 'dddd'; // "Saturday"}},// Computes a default event time formatting string if `timeFormat` is not explicitly
definedcomputeEventTimeFormat: function() {return this.view.opt('extraSmallTimeFormat'); // like "6p" or "6:30p"},// Computes a default `displayEventEnd` value if one is not explicltly definedcomputeDisplayEventEnd: function() {return this.colCnt == 1; // we'll likely have space if there's only one day},/* Cell System-------------------------
------------------------------------------------------------*/rangeUpdated: function() {var cellDates;var firstDay;var rowCnt;var colCnt;this.updateCellDates(); // populates cellDates and dayToCellOffsetscellDates = this.cellDates;if (this.breakOnWeeks) {// count columns until the day-of-week repeatsfirstDay = cellDates[0].day();for (colCnt = 1; colCnt <
cellDates.length; colCnt++) {if (cellDates[colCnt].day() == firstDay) {break;}}rowCnt = Math.ceil(cellDates.length / colCnt);}else {rowCnt = 1;colCnt = cellDates.length;}this.rowCnt = rowCnt;this.colCnt = colCnt;},// Populates cellDates and dayToCellOffsetsupdateCellDates: function() {var view = this.view;var date = this.start.clone();var dates =
[];var offset = -1;var offsets = [];while (date.isBefore(this.end)) { // loop each day from start to endif (view.isHiddenDay(date)) {offsets.push(offset + 0.5); // mark that it's between offsets}else {offset++;offsets.push(offset);dates.push(date.clone());}date.add(1, 'days');}this.cellDates = dates;this.dayToCellOffsets = offsets;},// Given a cell object,
generates its start date. Returns a reference-free copy.computeCellDate: function(cell) {var colCnt = this.colCnt;var index = cell.row * colCnt + (this.isRTL ? colCnt - cell.col - 1 : cell.col);return this.cellDates[index].clone();},// Retrieves the element representing the given rowgetRowEl: function(row) {return this.rowEls.eq(row);},// Retrieves the
element representing the given columngetColEl: function(col) {return this.dayEls.eq(col);},// Gets the whole-day element associated with the cellgetCellDayEl: function(cell) {return this.dayEls.eq(cell.row * this.colCnt + cell.col);},// Overrides Grid's method for when row coordinates are computedcomputeRowCoords: function() {var rowCoords =
Grid.prototype.computeRowCoords.call(this); // call the super-method// hack for extending last row (used by AgendaView)rowCoords[rowCoords.length - 1].bottom += this.bottomCoordPadding;return rowCoords;},/* Dates-----------------------------------------------------------------*/// Slices up a date range by row into an array of
segmentsrangeToSegs: function(range) {var isRTL = this.isRTL;var rowCnt = this.rowCnt;var colCnt = this.colCnt;var segs = [];var first, last; // inclusive cell-offset range for given rangevar row;var rowFirst, rowLast; // inclusive cell-offset range for current rowvar isStart, isEnd;var segFirst, segLast; // inclusive cell-offset range for segmentvar
seg;range = this.view.computeDayRange(range); // make whole-day range, considering nextDayThresholdfirst = this.dateToCellOffset(range.start);last = this.dateToCellOffset(range.end.subtract(1, 'days')); // offset of inclusive end datefor (row = 0; row < rowCnt; row++) {rowFirst = row * colCnt;rowLast = rowFirst + colCnt - 1;// intersect segment's
offset range with the row'ssegFirst = Math.max(rowFirst, first);segLast = Math.min(rowLast, last);// deal with in-between indicessegFirst = Math.ceil(segFirst); // in-between starts round to next cellsegLast = Math.floor(segLast); // in-between ends round to prev cellif (segFirst 0) {trEl.append('');}trEl.append(seg.el.attr('colspan', endCol - startCol));if
(endCol < colCnt) {trEl.append('');}this.bookendCells(trEl, type);return skeletonEl;}});;;/* Event-rendering methods for the DayGrid class--------------------------------------------------------*/DayGrid.mixin({rowStructs: null, // an array of objects, each holding information about a row's foreground event-rendering//
Unrenders all events currently rendered on the gridunrenderEvents: function() {this.removeSegPopover(); // removes the "more.." events popoverGrid.prototype.unrenderEvents.apply(this, arguments); // calls the super-method},// Retrieves all rendered segment objects currently rendered on the gridgetEventSegs: function() {return
Grid.prototype.getEventSegs.call(this) // get the segments from the super-method.concat(this.popoverSegs || []); // append the segments from the "more..." popover},// Renders the given background event segments onto the gridrenderBgSegs: function(segs) {// don't render timed background eventsvar allDaySegs = $.grep(segs, function(seg) {return
seg.event.allDay;});return Grid.prototype.renderBgSegs.call(this, allDaySegs); // call the super-method},// Renders the given foreground event segments onto the gridrenderFgSegs: function(segs) {var rowStructs;// render an `.el` on each seg// returns a subset of the segs. segs that were actually renderedsegs = this.renderFgSegEls(segs);rowStructs
= this.rowStructs = this.renderSegRows(segs);// append to each row's content skeletonthis.rowEls.each(function(i, rowNode) {$(rowNode).find('.fc-content-skeleton > table').append(rowStructs[i].tbodyEl);});return segs; // return only the segs that were actually rendered},// Unrenders all currently rendered foreground event
segmentsunrenderFgSegs: function() {var rowStructs = this.rowStructs || [];var rowStruct;while ((rowStruct = rowStructs.pop())) {rowStruct.tbodyEl.remove();}this.rowStructs = null;},// Uses the given events array to generate elements that should be appended to each row's content skeleton.// Returns an array of rowStruct objects (see the bottom
of `renderSegRow`).// PRECONDITION: each segment shoud already have a rendered and assigned `.el`renderSegRows: function(segs) {var rowStructs = [];var segRows;var row;segRows = this.groupSegRows(segs); // group into nested arrays// iterate each row of segment groupingsfor (row = 0; row < segRows.length; row++)
{rowStructs.push(this.renderSegRow(row, segRows[row]));}return rowStructs;},// Builds the HTML to be used for the default element for an individual segmentfgSegHtml: function(seg, disableResizing) {var view = this.view;var event = seg.event;var isDraggable = view.isEventDraggable(event);var isResizableFromStart = !disableResizing &&
event.allDay &&seg.isStart && view.isEventResizableFromStart(event);var isResizableFromEnd = !disableResizing && event.allDay &&seg.isEnd && view.isEventResizableFromEnd(event);var classes = this.getSegClasses(seg, isDraggable, isResizableFromStart || isResizableFromEnd);var skinCss = cssToStr(this.getEventSkinCss(event));var
timeHtml = '';var timeText;var titleHtml;classes.unshift('fc-day-grid-event', 'fc-h-event');// Only display a timed events time if it is the starting segmentif (seg.isStart) {timeText = this.getEventTimeText(event);if (timeText) {timeHtml = '' + htmlEscape(timeText) + '';}}titleHtml = '' +(htmlEscape(event.title || '') || ' ') + // we always want one line of
height'';return '' +'' +(this.isRTL ?titleHtml + ' ' + timeHtml : // put a natural space in betweentimeHtml + ' ' + titleHtml //) +'' +(isResizableFromStart ?'' :'') +(isResizableFromEnd ?'' :'') +'';},// Given a row # and an array of segments all in the same row, render a element, a skeleton that contains// the segments. Returns object with a bunch of
internal data about how the render was calculated.// NOTE: modifies rowSegsrenderSegRow: function(row, rowSegs) {var colCnt = this.colCnt;var segLevels = this.buildSegLevels(rowSegs); // group into sub-arrays of levelsvar levelCnt = Math.max(1, segLevels.length); // ensure at least one levelvar tbody = $('');var segMatrix = []; // lookup for which
segments are rendered into which level+col cellsvar cellMatrix = []; // lookup for all elements of the level+col matrixvar loneCellMatrix = []; // lookup for elements that only take up a single columnvar i, levelSegs;var col;var tr;var j, seg;var td;// populates empty cells from the current column (`col`) to `endCol`function emptyCellsUntil(endCol) {while
(col < endCol) {// try to grab a cell from the level above and extend its rowspan. otherwise, create a fresh celltd = (loneCellMatrix[i - 1] || [])[col];if (td) {td.attr('rowspan',parseInt(td.attr('rowspan') || 1, 10) + 1);}else {td = $('');tr.append(td);}cellMatrix[i][col] = td;loneCellMatrix[i][col] = td;col++;}}for (i = 0; i < levelCnt; i++) { // iterate through all
levelslevelSegs = segLevels[i];col = 0;tr = $('');segMatrix.push([]);cellMatrix.push([]);loneCellMatrix.push([]);// levelCnt might be 1 even though there are no actual levels. protect against this.// this single empty row is useful for styling.if (levelSegs) {for (j = 0; j < levelSegs.length; j++) { // iterate through segments in levelseg =
levelSegs[j];emptyCellsUntil(seg.leftCol);// create a container that occupies or more columns. append the event element.td = $('').append(seg.el);if (seg.leftCol != seg.rightCol) {td.attr('colspan', seg.rightCol - seg.leftCol + 1);}else { // a single-column segmentloneCellMatrix[i][col] = td;}while (col 1 and IE8, trEl.outerHeight() would return the height
of the largest cell,// so instead, find the tallest inner content element.trHeight = 0;trEl.find('> td > :first-child').each(iterInnerHeights);if (trEl.position().top + trHeight > rowHeight) {return i;}}return false; // should not limit at all},// Limits the given grid row to the maximum number of levels and injects "more" links if necessary.// `row` is the row
number.// `levelLimit` is a number for the maximum (inclusive) number of levels allowed.limitRow: function(row, levelLimit) {var _this = this;var rowStruct = this.rowStructs[row];var moreNodes = []; // array of "more" links and DOM nodesvar col = 0; // col #, left-to-right (not chronologically)var cell;var levelSegs; // array of segment objects in the
last allowable level, ordered left-to-rightvar cellMatrix; // a matrix (by level, then column) of all jQuery elements in the rowvar limitedNodes; // array of temporarily hidden level and segment DOM nodesvar i, seg;var segsBelow; // array of segment objects below `seg` in the current `col`var totalSegsBelow; // total number of segments below `seg` in
any of the columns `seg` occupiesvar colSegsBelow; // array of segment arrays, below seg, one for each column (offset from segs's first column)var td, rowspan;var segMoreNodes; // array of "more" cells that will stand-in for the current seg's cellvar j;var moreTd, moreWrap, moreLink;// Iterates through empty level cells and places "more" links inside
if need befunction emptyCellsUntil(endCol) { // goes from current `col` to `endCol`while (col < endCol) {cell = _this.getCell(row, col);segsBelow = _this.getCellSegs(cell, levelLimit);if (segsBelow.length) {td = cellMatrix[levelLimit - 1][col];moreLink = _this.renderMoreLink(cell, segsBelow);moreWrap =
$('').append(moreLink);td.append(moreWrap);moreNodes.push(moreWrap[0]);}col++;}}if (levelLimit && levelLimit < rowStruct.segLevels.length) { // is it actually over the limit?levelSegs = rowStruct.segLevels[levelLimit - 1];cellMatrix = rowStruct.cellMatrix;limitedNodes = rowStruct.tbodyEl.children().slice(levelLimit) // get level elements past
the limit.addClass('fc-limited').get(); // hide elements and get a simple DOM-nodes array// iterate though segments in the last allowable levelfor (i = 0; i < levelSegs.length; i++) {seg = levelSegs[i];emptyCellsUntil(seg.leftCol); // process empty cells before the segment// determine *all* segments below `seg` that occupy the same
columnscolSegsBelow = [];totalSegsBelow = 0;while (col = 0; i--) {labelInterval = moment.duration(AGENDA_STOCK_SUB_DURATIONS[i]);slotsPerLabel = divideDurationByDuration(labelInterval, slotDuration);if (isInt(slotsPerLabel) && slotsPerLabel > 1) {return labelInterval;}}return moment.duration(slotDuration); // fall back. clone},// Computes
a default column header formatting string if `colFormat` is not explicitly definedcomputeColHeadFormat: function() {if (this.colCnt > 1) { // multiple days, so full single date string WON'T be in title textreturn this.view.opt('dayOfMonthFormat'); // "Sat 12/10"}else { // single day, so full single date string will probably be in title textreturn 'dddd'; //
"Saturday"}},// Computes a default event time formatting string if `timeFormat` is not explicitly definedcomputeEventTimeFormat: function() {return this.view.opt('noMeridiemTimeFormat'); // like "6:30" (no AM/PM)},// Computes a default `displayEventEnd` value if one is not explicltly definedcomputeDisplayEventEnd: function() {return true;},/*
Cell System------------------------------------------------------------------------------------*/rangeUpdated: function() {var view = this.view;var colDates = [];var date;date = this.start.clone();while (date.isBefore(this.end)) {colDates.push(date.clone());date.add(1, 'day');date = view.skipHiddenDays(date);}if (this.isRTL) {colDates.reverse();}this.colDates
= colDates;this.colCnt = colDates.length;this.rowCnt = Math.ceil((this.maxTime - this.minTime) / this.snapDuration); // # of vertical snaps},// Given a cell object, generates its start date. Returns a reference-free copy.computeCellDate: function(cell) {var date = this.colDates[cell.col];var time = this.computeSnapTime(cell.row);date =
this.view.calendar.rezoneDate(date); // give it a 00:00 timedate.time(time);return date;},// Retrieves the element representing the given columngetColEl: function(col) {return this.dayEls.eq(col);},/* Dates----------------------------------------------------------------------*/// Given a row number of the grid, representing a "snap", returns a
time (Duration) from its start-of-daycomputeSnapTime: function(row) {return moment.duration(this.minTime + this.snapDuration * row);},// Slices up a date range by column into an array of segmentsrangeToSegs: function(range) {var colCnt = this.colCnt;var segs = [];var seg;var col;var colDate;var colRange;// normalize :(range = {start:
range.start.clone().stripZone(),end: range.end.clone().stripZone()};for (col = 0; col < colCnt; col++) {colDate = this.colDates[col]; // will be ambig time/timezonecolRange = {start: colDate.clone().time(this.minTime),end: colDate.clone().time(this.maxTime)};seg = intersectionToSeg(range, colRange); // both will be ambig timezoneif (seg) {seg.col =
col;segs.push(seg);}}return segs;},/* Coordinates-----------------------------------------------------------------*/updateSize: function(isResize) { // NOT a standard Grid methodthis.computeSlatTops();if (isResize) {this.updateSegVerticals();}},// Computes the top/bottom coordinates of each "snap" rowscomputeRowCoords: function()
{var originTop = this.el.offset().top;var items = [];var i;var item;for (i = 0; i < this.rowCnt; i++) {item = {top: originTop + this.computeTimeTop(this.computeSnapTime(i))};if (i > 0) {items[i - 1].bottom = item.top;}items.push(item);}item.bottom = item.top + this.computeTimeTop(this.computeSnapTime(i));return items;},// Computes the top
coordinate, relative to the bounds of the grid, of the given date.// A `startOfDayDate` must be given for avoiding ambiguity over how to treat midnight.computeDateTop: function(date, startOfDayDate) {return this.computeTimeTop(moment.duration(date.clone().stripZone() - startOfDayDate.clone().stripTime()));},// Computes the top coordinate,
relative to the bounds of the grid, of the given time (a Duration).computeTimeTop: function(time) {var slatCoverage = (time - this.minTime) / this.slotDuration; // floating-point value of # of slots coveredvar slatIndex;var slatRemainder;var slatTop;var slatBottom;// constrain. because minTime/maxTime might be customizedslatCoverage = Math.max(0,
slatCoverage);slatCoverage = Math.min(this.slatEls.length, slatCoverage);slatIndex = Math.floor(slatCoverage); // an integer index of the furthest whole slotslatRemainder = slatCoverage - slatIndex;slatTop = this.slatTops[slatIndex]; // the top position of the furthest whole slotif (slatRemainder) { // time spans part-way into the slotslatBottom =
this.slatTops[slatIndex + 1];return slatTop + (slatBottom - slatTop) * slatRemainder; // part-way between slots}else {return slatTop;}},// Queries each `slatEl` for its position relative to the grid's container and stores it in `slatTops`.// Includes the the bottom of the last slat as the last item in the array.computeSlatTops: function() {var tops = [];var
top;this.slatEls.each(function(i, node) {top = $(node).position().top;tops.push(top);});tops.push(top + this.slatEls.last().outerHeight()); // bottom of the last slatthis.slatTops = tops;},/* Event Drag Visualization---------------------------------------------------------*/// Renders a visual indication of an event being dragged over the
specified date(s).// dropLocation's end might be null, as well as `seg`. See Grid::renderDrag for more info.// A returned value of `true` signals that a mock "helper" event has been rendered.renderDrag: function(dropLocation, seg) {if (seg) { // if there is event information for this drag, render a helper eventthis.renderRangeHelper(dropLocation,
seg);this.applyDragOpacity(this.helperEl);return true; // signal that a helper has been rendered}else {// otherwise, just render a highlightthis.renderHighlight(this.eventRangeToSegs(dropLocation));}},// Unrenders any visual indication of an event being draggedunrenderDrag: function() {this.unrenderHelper();this.unrenderHighlight();},/* Event
Resize Visualization----------------------------------------------------------*/// Renders a visual indication of an event being resizedrenderEventResize: function(range, seg) {this.renderRangeHelper(range, seg);},// Unrenders any visual indication of an event being resizedunrenderEventResize: function()
{this.unrenderHelper();},/* Event Helper-------------------------------------------------------*/// Renders a mock "helper" event. `sourceSeg` is the original segment object and might be null (an external drag)renderHelper: function(event, sourceSeg) {var segs = this.eventsToSegs([ event ]);var tableEl;var i, seg;var
sourceEl;segs = this.renderFgSegEls(segs); // assigns each seg's el and returns a subset of segs that were renderedtableEl = this.renderSegTable(segs);// Try to make the segment that is in the same row as sourceSeg look the samefor (i = 0; i < segs.length; i++) {seg = segs[i];if (sourceSeg && sourceSeg.col === seg.col) {sourceEl =
sourceSeg.el;seg.el.css({left: sourceEl.css('left'),right: sourceEl.css('right'),'margin-left': sourceEl.css('margin-left'),'margin-right': sourceEl.css('margin-right')});}}this.helperEl = $('').append(tableEl).appendTo(this.el);},// Unrenders any mock helper eventunrenderHelper: function() {if (this.helperEl) {this.helperEl.remove();this.helperEl = null;}},/*
Selection-------------------------------------------------------*/// Renders a visual indication of a selection. Overrides the default, which was to simply render a highlight.renderSelection: function(range) {if (this.view.opt('selectHelper')) { // this setting signals that a mock helper event should be
renderedthis.renderRangeHelper(range);}else {this.renderHighlight(this.selectionRangeToSegs(range));}},// Unrenders any visual indication of a selectionunrenderSelection: function() {this.unrenderHelper();this.unrenderHighlight();},/* Fill System (highlight, background events, business hours)-----------------
-------------------------------*/// Renders a set of rectangles over the given time segments.// Only returns segments that successfully rendered.renderFill: function(type, segs, className) {var segCols;var skeletonEl;var trEl;var col, colSegs;var tdEl;var containerEl;var dayDate;var i, seg;if (segs.length) {segs = this.renderFillSegEls(type, segs); // assignes
`.el` to each seg. returns successfully rendered segssegCols = this.groupSegCols(segs); // group into sub-arrays, and assigns 'col' to each segclassName = className || type.toLowerCase();skeletonEl = $('' +'' +'');trEl = skeletonEl.find('tr');for (col = 0; col < segCols.length; col++) {colSegs = segCols[col];tdEl = $('').appendTo(trEl);if (colSegs.length)
{containerEl = $('').appendTo(tdEl);dayDate = this.colDates[col];for (i = 0; i < colSegs.length; i++) {seg = colSegs[i];containerEl.append(seg.el.css({top: this.computeDateTop(seg.start, dayDate),bottom: -this.computeDateTop(seg.end, dayDate) // the y position of the bottom edge}));}}}this.bookendCells(trEl,
type);this.el.append(skeletonEl);this.elsByFill[type] = skeletonEl;}return segs;}});;;/* Event-rendering methods for the TimeGrid class----------------------------------------------------*/TimeGrid.mixin({eventSkeletonEl: null, // has cells with event-containers, which contain absolutely positioned event elements// Renders
the given foreground event segments onto the gridrenderFgSegs: function(segs) {segs = this.renderFgSegEls(segs); // returns a subset of the segs. segs that were actually renderedthis.el.append(this.eventSkeletonEl = $('').append(this.renderSegTable(segs)));return segs; // return only the segs that were actually rendered},// Unrenders all currently
rendered foreground event segmentsunrenderFgSegs: function(segs) {if (this.eventSkeletonEl) {this.eventSkeletonEl.remove();this.eventSkeletonEl = null;}},// Renders and returns the portion of the event-skeleton.// Returns an object with properties 'tbodyEl' and 'segs'.renderSegTable: function(segs) {var tableEl = $('');var trEl =
tableEl.find('tr');var segCols;var i, seg;var col, colSegs;var containerEl;segCols = this.groupSegCols(segs); // group into sub-arrays, and assigns 'col' to each segthis.computeSegVerticals(segs); // compute and assign top/bottomfor (col = 0; col < segCols.length; col++) { // iterate each column groupingcolSegs =
segCols[col];this.placeSlotSegs(colSegs); // compute horizontal coordinates, z-index's, and reorder the arraycontainerEl = $('');// assign positioning CSS and insert into containerfor (i = 0; i < colSegs.length; i++) {seg = colSegs[i];seg.el.css(this.generateSegPositionCss(seg));// if the height is short, add a className for alternate stylingif (seg.bottom -
seg.top < 30) {seg.el.addClass('fc-short');}containerEl.append(seg.el);}trEl.append($('').append(containerEl));}this.bookendCells(trEl, 'eventSkeleton');return tableEl;},// Given an array of segments that are all in the same column, sets the backwardCoord and forwardCoord on each.// NOTE: Also reorders the given array by date!placeSlotSegs:
function(segs) {var levels;var level0;var i;this.sortSegs(segs); // order by datelevels = buildSlotSegLevels(segs);computeForwardSlotSegs(levels);if ((level0 = levels[0])) {for (i = 0; i < level0.length; i++) {computeSlotSegPressures(level0[i]);}for (i = 0; i < level0.length; i++) {this.computeSlotSegCoords(level0[i], 0, 0);}}},// Calculate
seg.forwardCoord and seg.backwardCoord for the segment, where both values range// from 0 to 1. If the calendar is left-to-right, the seg.backwardCoord maps to "left" and// seg.forwardCoord maps to "right" (via percentage). Vice-versa if the calendar is right-to-left.//// The segment might be part of a "series", which means consecutive segments with
the same pressure// who's width is unknown until an edge has been hit. `seriesBackwardPressure` is the number of// segments behind this one in the current series, and `seriesBackwardCoord` is the starting// coordinate of the first segment in the series.computeSlotSegCoords: function(seg, seriesBackwardPressure, seriesBackwardCoord) {var
forwardSegs = seg.forwardSegs;var i;if (seg.forwardCoord === undefined) { // not already computedif (!forwardSegs.length) {// if there are no forward segments, this segment should butt up against the edgeseg.forwardCoord = 1;}else {// sort highest pressure firstthis.sortForwardSlotSegs(forwardSegs);// this segment's forwardCoord will be
calculated from the backwardCoord of the// highest-pressure forward segment.this.computeSlotSegCoords(forwardSegs[0], seriesBackwardPressure + 1, seriesBackwardCoord);seg.forwardCoord = forwardSegs[0].backwardCoord;}// calculate the backwardCoord from the forwardCoord. consider the seriesseg.backwardCoord = seg.forwardCoord -
(seg.forwardCoord - seriesBackwardCoord) / // available width for series(seriesBackwardPressure + 1); // # of segments in the series// use this segment's coordinates to computed the coordinates of the less-pressurized// forward segmentsfor (i=0; i "6pm" / "6:30pm"smallTimeFormat: function(momOptions) {return
momOptions.longDateFormat('LT').replace(':mm', '(:mm)').replace(/(\Wmm)$/, '($1)') // like above, but for foreign langs.replace(/\s*a$/i, 'a'); // convert AM/PM/am/pm to lowercase. remove any spaces beforehand},// Produces format strings like "h(:mm)t" -> "6p" / "6:30p"extraSmallTimeFormat: function(momOptions) {return
momOptions.longDateFormat('LT').replace(':mm', '(:mm)').replace(/(\Wmm)$/, '($1)') // like above, but for foreign langs.replace(/\s*a$/i, 't'); // convert to AM/PM/am/pm to lowercase one-letter. remove any spaces beforehand},// Produces format strings like "ha" / "H" -> "6pm" / "18"hourFormat: function(momOptions) {return
momOptions.longDateFormat('LT').replace(':mm', '').replace(/(\Wmm)$/, '') // like above, but for foreign langs.replace(/\s*a$/i, 'a'); // convert AM/PM/am/pm to lowercase. remove any spaces beforehand},// Produces format strings like "h:mm" -> "6:30" (with no AM/PM)noMeridiemTimeFormat: function(momOptions) {return
momOptions.longDateFormat('LT').replace(/\s*a$/i, ''); // remove trailing AM/PM}};// options that should be computed off live calendar options (considers override options)// TODO: best place for this? related to lang?// TODO: flipping text based on isRTL is a bad idea because the CSS `direction` might want to handle itvar instanceComputableOptions
= {// Produces format strings for results like "Mo 16"smallDayDateFormat: function(options) {return options.isRTL ?'D dd' :'dd D';},// Produces format strings for results like "Wk 5"weekFormat: function(options) {return options.isRTL ?'w[ ' + options.weekNumberTitle + ']' :'[' + options.weekNumberTitle + ' ]w';},// Produces format strings for results
like "Wk5"smallWeekFormat: function(options) {return options.isRTL ?'w[' + options.weekNumberTitle + ']' :'[' + options.weekNumberTitle + ']w';}};function populateInstanceComputableOptions(options) {$.each(instanceComputableOptions, function(name, func) {if (options[name] == null) {options[name] = func(options);}});}// Returns moment's
internal locale data. If doesn't exist, returns English.// Works with moment-pre-2.8function getMomentLocaleData(langCode) {var func = moment.localeData || moment.langData;return func.call(moment, langCode) ||func.call(moment, 'en'); // the newer localData could return null, so fall back to en}// Initialize English by forcing computation of
moment-derived options.// Also, sets it as the default.fc.lang('en', Calendar.englishDefaults);;;/* Top toolbar area with buttons and title-------------------------------------------------------*/// TODO: rename all header-related things to "toolbar"function Header(calendar, options) {var t = this;// exportst.render =
render;t.removeElement = removeElement;t.updateTitle = updateTitle;t.activateButton = activateButton;t.deactivateButton = deactivateButton;t.disableButton = disableButton;t.enableButton = enableButton;t.getViewsWithButtons = getViewsWithButtons;// localsvar el = $();var viewsWithButtons = [];var tm;function render() {var sections =
options.header;tm = options.theme ? 'ui' : 'fc';if (sections) {el = $("").append(renderSection('left')).append(renderSection('right')).append(renderSection('center')).append('');return el;}}function removeElement() {el.remove();el = $();}function renderSection(position) {var sectionEl = $('');var buttonStr = options.header[position];if (buttonStr)
{$.each(buttonStr.split(' '), function(i) {var groupChildren = $();var isOnlyButtons = true;var groupEl;$.each(this.split(','), function(j, buttonName) {var customButtonProps;var viewSpec;var buttonClick;var overrideText; // text explicitly set by calendar's constructor options. overcomes iconsvar defaultText;var themeIcon;var normalIcon;var
innerHtml;var classes;var button; // the elementif (buttonName == 'title') {groupChildren = groupChildren.add($(' ')); // we always want it to take up heightisOnlyButtons = false;}else {if ((customButtonProps = (calendar.options.customButtons || {})[buttonName])) {buttonClick = function(ev) {if (customButtonProps.click)
{customButtonProps.click.call(button[0], ev);}};overrideText = ''; // icons will override textdefaultText = customButtonProps.text;}else if ((viewSpec = calendar.getViewSpec(buttonName))) {buttonClick = function() {calendar.changeView(buttonName);};viewsWithButtons.push(buttonName);overrideText = viewSpec.buttonTextOverride;defaultText
= viewSpec.buttonTextDefault;}else if (calendar[buttonName]) { // a calendar methodbuttonClick = function() {calendar[buttonName]();};overrideText = (calendar.overrides.buttonText || {})[buttonName];defaultText = options.buttonText[buttonName]; // everything else is considered default}if (buttonClick) {themeIcon =customButtonProps ?
customButtonProps.themeIcon :options.themeButtonIcons[buttonName];normalIcon =customButtonProps ?customButtonProps.icon :options.buttonIcons[buttonName];if (overrideText) {innerHtml = htmlEscape(overrideText);}else if (themeIcon && options.theme) {innerHtml = "";}else if (normalIcon && !options.theme) {innerHtml = "";}else
{innerHtml = htmlEscape(defaultText);}classes = ['fc-' + buttonName + '-button',tm + '-button',tm + '-state-default'];button = $( // type="button" so that it doesn't submit a form'' +innerHtml +'').click(function(ev) {// don't process clicks for disabled buttonsif (!button.hasClass(tm + '-state-disabled')) {buttonClick(ev);// after the click action, if the
button becomes the "active" tab, or disabled,// it should never have a hover class, so remove it now.if (button.hasClass(tm + '-state-active') ||button.hasClass(tm + '-state-disabled')) {button.removeClass(tm + '-state-hover');}}}).mousedown(function() {// the *down* effect (mouse pressed in).// only on buttons that are not the "active" tab, or
disabledbutton.not('.' + tm + '-state-active').not('.' + tm + '-state-disabled').addClass(tm + '-state-down');}).mouseup(function() {// undo the *down* effectbutton.removeClass(tm + '-state-down');}).hover(function() {// the *hover* effect.// only on buttons that are not the "active" tab, or disabledbutton.not('.' + tm + '-state-active').not('.' + tm + '-state-
disabled').addClass(tm + '-state-hover');},function() {// undo the *hover* effectbutton.removeClass(tm + '-state-hover').removeClass(tm + '-state-down'); // if mouseleave happens before mouseup});groupChildren = groupChildren.add(button);}}});if (isOnlyButtons) {groupChildren.first().addClass(tm + '-corner-left').end().last().addClass(tm + '-corner-
right').end();}if (groupChildren.length > 1) {groupEl = $('');if (isOnlyButtons) {groupEl.addClass('fc-button-group');}groupEl.append(groupChildren);sectionEl.append(groupEl);}else {sectionEl.append(groupChildren); // 1 or 0 children}});}return sectionEl;}function updateTitle(text) {el.find('h2').text(text);}function activateButton(buttonName)
{el.find('.fc-' + buttonName + '-button').addClass(tm + '-state-active');}function deactivateButton(buttonName) {el.find('.fc-' + buttonName + '-button').removeClass(tm + '-state-active');}function disableButton(buttonName) {el.find('.fc-' + buttonName + '-button').attr('disabled', 'disabled').addClass(tm + '-state-disabled');}function
enableButton(buttonName) {el.find('.fc-' + buttonName + '-button').removeAttr('disabled').removeClass(tm + '-state-disabled');}function getViewsWithButtons() {return viewsWithButtons;}};fc.sourceNormalizers = [];fc.sourceFetchers = [];var ajaxDefaults = {dataType: 'json',cache: false};var eventGUID = 1;function EventManager(options) { //
assumed to be a calendarvar t = this;// exportst.isFetchNeeded = isFetchNeeded;t.fetchEvents = fetchEvents;t.addEventSource = addEventSource;t.removeEventSource = removeEventSource;t.updateEvent = updateEvent;t.renderEvent = renderEvent;t.removeEvents = removeEvents;t.clientEvents = clientEvents;t.mutateEvent =
mutateEvent;t.normalizeEventRange = normalizeEventRange;t.normalizeEventRangeTimes = normalizeEventRangeTimes;t.ensureVisibleEventRange = ensureVisibleEventRange;// importsvar reportEvents = t.reportEvents;// localsvar stickySource = { events: [] };var sources = [ stickySource ];var rangeStart, rangeEnd;var currentFetchID = 0;var
pendingSourceCnt = 0;var cache = []; // holds events that have already been expanded$.each((options.events ? [ options.events ] : []).concat(options.eventSources || []),function(i, sourceInput) {var source = buildEventSource(sourceInput);if (source) {sources.push(source);}});/* Fetching--------------------------------------------------------*/function
isFetchNeeded(start, end) {return !rangeStart || // nothing has been fetched yet?// or, a part of the new range is outside of the old range? (after normalizing)start.clone().stripZone() < rangeStart.clone().stripZone() ||end.clone().stripZone() > rangeEnd.clone().stripZone();}function fetchEvents(start, end) {rangeStart = start;rangeEnd = end;cache =
[];var fetchID = ++currentFetchID;var len = sources.length;pendingSourceCnt = len;for (var i=0; i
```