prosemirror-transform 0.21.1 (2017-05-16)

Bug fixes

`addMark` no longer assumes marks always exclude only themselves.

`replaceRange`](http://prosemirror.net/docs/ref/version/0.21.0.html#transform.Transform.replaceRange) and `deleteRange` will no longer expand the range across isolating node boundaries.

prosemirror-view 0.21.1 (2017-05-09)

Bug fixes

Copying and pasting table cells on Edge no longer strips the table structure.

prosemirror-model 0.21.0 (2017-05-03)

Breaking changes

The `openLeft` and `openRight` properties of `Slice` objects have been renamed to `openStart` and `openEnd` to avoid confusion in right-to-left text. The old names will continue to work with a warning until the next release.

New features

Mark serializing functions now get a second parameter that indicates whether the mark's content is inline or block nodes.

Setting a mark serializer to `null` in a `DOMSerializer` can now be used to omit that mark when serializing.

Node specs support a new property `isolating`, which is used to disable editing actions like backspacing and lifting across such a node's boundaries.

prosemirror-state 0.21.0 (2017-05-03)

Breaking changes

`Selection.atStart`, and `atEnd` no longer take a second `textOnly` parameter.

New features

`Selection.near`, `atStart`, and `atEnd` will now fall back to returning an `AllSelection` when unable to find a valid selection. This removes the (undocumented) requirement that documents always contain a valid selection position (though you'll probably still want to maintain this for practical UI reasons).

prosemirror-view 0.21.0 (2017-05-03)

Breaking changes

The `associative` option to widget decorations is no longer supported. To make a widget left-associative, set its `side` option to a negative number. `associative` will continue to work with a warning until the next release.

New features

Widget decorations now support a `side` option that controls which side of them the cursor is drawn, where they move when content is inserted at their position, and the order in which they appear relative to other widgets at the same position.

prosemirror-view 0.20.5 (2017-05-02)

Bug fixes

Fixes an issue where the DOM selection could be shown on the wrong side of hard break or image nodes.

prosemirror-view 0.20.4 (2017-04-24)

Bug fixes

Fix a bug that prevented the DOM selection from being updated when the new position was near the old one in some circumstances.

Stop interfering with alt-d keypresses on OS X.

Fix issue where reading a DOM change in a previously empty node could crash.

Fixes crash when reading a change that removed a decorated text node from the DOM.

prosemirror-view 0.20.3 (2017-04-12)

Bug fixes

Shift-pasting and pasting into a code block now does the right thing on IE and Edge.

prosemirror-view 0.20.2 (2017-04-05)

Bug fixes

Fixes a bug that broke dragging from the editor.

prosemirror-view 0.20.1 (2017-04-04)

Bug fixes

Typing in code blocks no longer replaces newlines with spaces.

Copy and paste on Internet Explorer, Edge, and mobile Safari should now behave more like it does on other browsers. Handlers are called, and the changes to the document are made by ProseMirror's code, not the browser.

Fixes a problem where triple-clicking the editor would sometimes cause the scroll position to inexplicably jump around on IE11.

prosemirror-model 0.20.0 (2017-04-03)

Breaking changes

Newlines in the text are now normalized to spaces when parsing except when you set `preserveWhitespace` to `"full"` in your options or in a parse rule.

Bug fixes

Fix crash in IE when parsing DOM content.

New features

Fragments now have `nodesBetween` and `descendants` methods, providing the same functionality as the methods by the same name on nodes.

Resolved positions now have `max` and `min` methods to easily find a maximum or minimum position.

prosemirror-transform 0.20.0 (2017-04-03)

Bug fixes

Fixes issue where replacing would sometimes unexpectedly split nodes.

prosemirror-state 0.20.0 (2017-04-03)

Breaking changes

`Selection.near` no longer accepts a `textOnly` parameter.

Bug fixes

`TextSelection.between` may now return a node selection when the document does not contain a valid cursor position.

New features

`Selection` objects now implement a `content` method that returns their content. This is used to determine what ends up on the clipboard when the selection is copied or dragged.

Selections may now specify multiple ranges that they cover, to generalize to more types of selections. The `Selection` superclass constructor takes an array of ranges as optional third argument.

Selections gained new methods `replace` and `replaceWith` to provide subclasses more control over how selections of that type respond to being deleted or overwritten.

Selections have a new method `getBookmark` that custom selection classes can implement to allow the undo history to accurately store and restore them.

The new selection class `AllSelection` can be used to select the entire document.

prosemirror-view 0.20.0 (2017-04-03)

Breaking changes

The `inclusiveLeft` and `inclusiveRight` options to inline decorations were renamed to `inclusiveStart` and `inclusiveEnd` so that they also make sense in right-to-left text. The old names work with a warning until the next release.

The default styling for lists and blockquotes was removed from `prosemirror.css`. (They were moved to the `example-setup` module.)

Bug fixes

Fixes reading of selection in Chrome in a shadow DOM.

Registering DOM event handlers that the editor doesn't listen to by default with the `handleDOMEvents` prop should work again.

Backspacing after turning off a mark now works again in Firefox.

New features

The new props `handlePaste` and `handleDrop` can be used to override drop and paste behavior.

prosemirror-inputrules 0.20.0 (2017-04-03)

Breaking changes

The input rules plugin no longer implicitly binds backspace to undo the last applied rule.

New features

This module now exposes a command `undoInputRule`, which will revert an input rule when run directly after one was applied.

prosemirror-history 0.20.0 (2017-04-03)

Bug fixes

Appended transactions no longer generate undo history events.

prosemirror-commands 0.20.0 (2017-04-03)

New features

The new `selectAll` command, bound to Mod-a in the base keymap, sets the selection to an `AllSelection`.

prosemirror-schema-list 0.20.0 (2017-04-03)

New features

The `liftListItem` command can now lift items out of a list entirely, when the parent node isn't another list.

prosemirror-view 0.19.1 (2017-03-18)

Bug fixes

Fixes a number of issues with characters being duplicated or disappearing when typing on mark boundaries.

prosemirror-state 0.19.1 (2017-03-17)

Bug fixes

Fix an issue where `ensureMarks` would fail to reset the marks to the empty set when turning off the last mark.

prosemirror-model 0.19.0 (2017-03-16)

Breaking changes

`MarkSpec.inclusiveRight` was replaced by `inclusive`, which behaves slightly differently. `inclusiveRight` will be interpreted as `inclusive` (with a warning) until the next release.

New features

The new `inlineContent` property on nodes and node types tells you whether a node type supports inline content.

`MarkSpec.inclusive` can now be used to control whether content inserted at the boundary of a mark receives that mark.

Parse rule `context` restrictions can now use node groups, not just node names, to specify valid context.

prosemirror-state 0.19.0 (2017-03-16)

Breaking changes

`Selection.between` is now called `TextSelection.between`, and only returns text selections.

The JSON representation of selections changed. `fromJSON` will continue to understand the old representation, but if your own code touches the JSON data, you'll have to adjust it.

All `Selection` objects now have `$head`/`$anchor` properties, so those can no longer be used to recognize text selections (use `$cursor` or `instanceof`).

New features

It is now possible to write your own `Selection` subclasses and set the editor selection to an instance of them (provided you implement all required methods and register them with `Selection.jsonID`).

Text selections now have a `$cursor` getter which returns a position only if this is a cursor selection.

The new `Transaction.ensureMarks` method makes it easier to ensure given set of active marks without needlessly setting `storedMarks`.

prosemirror-view 0.19.0 (2017-03-16)

Breaking changes

`endOfTextblock` no longer always returns false for horizontal motion on non-cursor selections, but checks the position of the selection head instead.

Bug fixes

Typing after adding/removing a mark no longer briefly shows the new text with the wrong marks.

`posAtCoords` is now more reliable on modern browsers by using browser APIs.

Fix a bug where the view would in some circumstances leave superfluous DOM nodes around inside marks.

New features

You can now override the selection the editor creates for a given DOM selection with the `createSelectionBetween` prop.

prosemirror-history 0.19.0 (2017-03-16)

New features

A new function `closeHistory` can be used to force separation of history events at the start of a given transaction.

prosemirror-collab 0.19.0 (2017-03-16)

New features

You can now use strings (as well as numbers) as client IDs (this already worked, but now the documentation reflects this).

prosemirror-commands 0.19.0 (2017-03-16)

Bug fixes

Calling `joinBackward` at the start of a node that can't be joined no longer raises an error.

prosemirror-schema-basic 0.19.0 (2017-03-16)

Breaking changes

Link marks are now non-inclusive by default.

prosemirror-model 0.18.0 (2017-02-24)

Breaking changes

`schema.nodeSpec` and `schema.markSpec` have been deprecated in favor of `schema.spec`. The properties still work with a warning in this release, but will be dropped in the next.

New features

`Node` objects now have a `check` method which can be used to assert that they conform to the schema.

Node specs now support an `atom` property, and nodes an `isAtom` accessor, which is currently only used to determine whether such nodes should be directly selectable (for example when they are rendered as an uneditable node view).

The new `excludes` field on mark specs can be used to control the marks that this mark may coexist with. Mark type objects also gained an `excludes` *method* to querty this relation.

Mark specs now support a `group` property, and marks can be referred to by group name in content specs.

The `Schema` class now provides its whole spec under its `spec` property.

The name of a schema's default top-level node is now configurable. You can use `schema.topNodeType` to retrieve the top node type.

Parse rules now support a `context` field that can be used to only make the rule match inside certain ancestor nodes.

prosemirror-transform 0.18.0 (2017-02-24)

New features

`Transform.setNodeType` now takes an optional argument to set the new node's attributes.

Steps now provide an `offset` method, which makes it possible to create a copy the step with its position offset by a given amount.

`docChanged` is now a property on the `Transform` class, rather than its `Transaction` subclass.

`Mapping` instances now have `invert` and `appendMappingInverted` methods to make mapping through them in reverse easier.

prosemirror-state 0.18.0 (2017-02-24)

Breaking changes

Plugin objects now store their spec under a `spec` instead of an `options` property. The `options` property still works with a warning in this release.

prosemirror-view 0.18.0 (2017-02-24)

Breaking changes

`Decoration` objects now store their definition object under `spec`, not `options`. The old property name still works, with a warning, until the next release.

Bug fixes

Fix bug where calling `focus` when there was a text selection would sometimes result in `state.selection` receiving an incorrect value.

`EditorView.props` now has its `state` property updated when you call `updateState`.

Putting decorations on or inside a node view with an `update` method now works.

New features

Plugin view update methods are now passed the view's previous state as second argument.

The `place` agument to the `EditorView` constructor can now be an object with a `mount` property to directly provide the node that should be made editable.

The new `EditorView.setProps` method makes it easier to update individual props.

prosemirror-keymap 0.18.0 (2017-02-24)

New features

Add a `keydownHandler` function, which takes a keymap and produces a `handleKeydown` prop-style function.

prosemirror-history 0.18.0 (2017-02-24)

Bug fixes

Fix a problem where simultaneous collaborative editing could break the undo history.

prosemirror-collab 0.18.0 (2017-02-24)

New features

`sendableSteps` now also returns information about the original transactions that produced the steps.

prosemirror-commands 0.18.0 (2017-02-24)

New features

New command `splitBlockKeepMarks` which splits a block but preserves the marks at the cursor.

prosemirror-view 0.17.7 (2017-02-08)

Bug fixes

Fixes crash in the code that maintains the scroll position when the document is empty or hidden.

prosemirror-view 0.17.6 (2017-02-08)

Bug fixes

Transactions that shouldn't scroll the selection into view now no longer do so.

prosemirror-state 0.17.1 (2017-02-08)

Bug fixes

`Transaction.scrolledIntoView` no longer always returns true.

`Selection.near` now takes a third `textOnly` argument, as the docs already claimed.

prosemirror-menu 0.17.1 (2017-02-07)

Bug fixes

Fix method call to method that doesn't exist on IE11.

prosemirror-view 0.17.4 (2017-02-02)

Bug fixes

Fixes bug where widget decorations would sometimes get parsed as content when editing near them.

The editor now prevents the behavior of Ctrl-d and Ctrl-h on textblock boundaries on OS X, as intended.

Make sure long words don't cause a horizontal scrollbar in Firefox

Various behavior fixes for IE11.

prosemirror-history 0.17.1 (2017-02-02)

Bug fixes

Fix issue where collaborative editing corner cases could corrupt the history.

prosemirror-view 0.17.3 (2017-01-19)

Bug fixes

DOM changes deleting a node's inner wrapping DOM element (for example the `<code>` tag in a schema-basic code block) no longer break the editor.

prosemirror-view 0.17.2 (2017-01-16)

Bug fixes

Call custom click handlers before applying select-node behavior for a ctrl/cmd-click.

Fix failure to apply DOM changes that start at document position 0.

prosemirror-commands 0.17.1 (2017-01-16)

Bug fixes

Make sure `toggleMark` also works in the top-level node (when it is a textblock).

prosemirror-view 0.17.1 (2017-01-07)

Bug fixes

Fix issue where a document update that left the selection in the same place sometimes led to an incorrect DOM selection.

Make sure `EditorView.focus` doesn't cause the browser to scroll the top of the editor into view.

prosemirror-model 0.17.0 (2017-01-05)

Breaking changes

`Node.marksAt` was replaced with `ResolvedPos.marks`. It still works (with a warning) in this release, but will be removed in the next one.

prosemirror-state 0.17.0 (2017-01-05)

Breaking changes

The way state is updated was changed. Instead of applying an action (a raw object with a `type` property), it is now done by applying a `Transaction`.

The `EditorTransform` class was renamed `Transaction`, and extended to allow changing the set of stored marks and attaching custom metadata.

New features

Plugins now accept a `filterTransaction` option that can be used to filter out transactions as they come in.

Plugins also got an `appendTransaction` option making it possible to follow up transactions with another transaction.

prosemirror-view 0.17.0 (2017-01-05)

Breaking changes

The `handleDOMEvent` prop has been dropped in favor of the `handleDOMEvents` (plural) prop.

The `onChange` prop has been replaced by a `dispatchTransaction` prop (which takes a transaction instead of an action).

New features

Added support for a `handleDOMEvents` prop, which allows you to provide handler functions per DOM event, and works even for events that the editor doesn't normally add a handler for.

Add view method `dispatch`, which provides a convenient way to dispatch transactions.

The `dispatchTransaction` (used to be `onAction`) prop is now optional, and will default to simply applying the transaction to the current view state.

Widget decorations now accept an option `associative` which can be used to configure on which side of content inserted at their position they end up.

Typing immediately after deleting text now preserves the marks of the deleted text.

Transactions that update the selection because of mouse or touch input now get a metadata property `pointer` with the value `true`.

prosemirror-commands 0.17.0 (2017-01-05)

Breaking changes

The `dispatch` function passed to commands is now passed a `Transaction`, not an action object.

prosemirror-view 0.16.0 (2016-12-23)

Bug fixes

Solve problem where setting a node selection would trigger a DOM read, leading to the selection being reset.

prosemirror-state 0.16.0 (2016-12-23)

New features

Plugins now take a `view` option that can be used to interact with the editor view.

prosemirror-view 0.16.0 (2016-12-23)

Breaking changes

The `spellcheck`, `label`, and `class` props are now replaced by an `attributes` prop.

Bug fixes

Ignoring/aborting an action should no longer lead to the DOM being stuck in an outdated state.

Typing at the end of a textblock which ends in a non-text node now actually works.

DOM nodes for leaf document nodes are now set as non-editable to prevent various issues such as stray cursors inside of them and Firefox adding image resize controls.

Inserting a node no longer causes nodes of the same type after it to be neednessly redrawn.

New features

Add a new editor prop `editable` which controls whether the editor's `contentEditable` behavior is enabled.

Plugins and props can now set any DOM attribute on the outer editor node using the `attributes` prop.

Node view constructors and update methods now have access to the node's wrapping decorations, which can be used to pass information to a node view without encoding it in the document.

Attributes added or removed by node and inline decorations no longer cause the nodes inside of them to be fully redrawn, making node views more stable and allowing CSS transitions to be used.

prosemirror-view 0.15.2 (2016-12-10)

Bug fixes

The native selection is now appropriately hidden when there is a node selection.

prosemirror-view 0.15.1 (2016-12-10)

Bug fixes

Fix DOM parsing for decorated text nodes.

prosemirror-model 0.15.0 (2016-12-10)

Breaking changes

`ResolvedPos.atNodeBoundary` is deprecated and will be removed in the next release. Use `textOffset > 0` instead.

New features

Parse rules associated with a schema can now specify a `priority` to influence the order in which they are applied.

Resolved positions have a new getter `textOffset` to find their position within a text node (if any).

prosemirror-transform 0.15.0 (2016-12-10)

Bug fixes

Fix bug where pasted/inserted content would sometimes get incorrectly closed at the right side.

prosemirror-state 0.15.0 (2016-12-10)

Breaking changes

Selection actions no longer scroll the new selection into view by default (they never were supposed to, but due to a bug they did). Add a `scrollIntoView` property to the action to get this behavior.

prosemirror-view 0.15.0 (2016-12-10)

Breaking changes

The editor view no longer wraps its editable DOM element in a wrapper element. The `ProseMirror` CSS class now applies directly to the editable element. The `ProseMirror-content` CSS class is still present for ease of upgrading but will be dropped in the next release.

The editor view no longer draws a drop cursor when dragging content over the editor. The new `prosemirror-dropcursor` module implements this as a plugin.

Bug fixes

Simple typing and backspacing now gets handled by the browser without ProseMirror redrawing the touched nodes, making spell-checking and various platform-specific input tricks (long-press on OS X, double space on iOS) work in the editor.

Improve tracking of DOM nodes that have been touched by user changes, so that `updateState` can reliably fix them.

Changes to the document that happen while dragging editor content no longer break moving of the content.

Adding or removing a mark directly in the DOM (for example with the bold/italic buttons in iOS' context menu) now produces mark steps, rather than replace steps.

Pressing backspace at the start of a paragraph on Android now allows key handlers for backspace to fire.

Toggling a mark when there is no selection now works better on mobile platforms.

New features

Introduces an `endOfTextblock` method on views, which can be used to find out in a bidi- and layout-aware way whether the selection is on the edge of a textblock.

prosemirror-commands 0.15.0 (2016-12-10)

Breaking changes

Drops suppport for `delete(Char|Word)(Before|After)` and `move(Back|For)ward`, since we are now letting the browser handle those natively.

Bug fixes

The `joinForward` and `joinBackward` commands can now strip out markup and nodes that aren't allowed in the joined node.

New features

A new command `exitCode` allows a user to exit a code block by creating a new paragraph below it.

The `joinForward` and `joinBackward` commands now use a bidirectional-text-aware way to determine whether the cursor is at the proper side of its parent textblock when they are passed a view.

prosemirror-view 0.14.4 (2016-12-02)

Bug fixes

Fix issue where node decorations would stick around in the DOM after the decoration was removed.

Setting or removing a node selection in an unfocused editor now properly updates the DOM to show that selection.

prosemirror-model 0.14.1 (2016-11-30)

Bug fixes

`DOMParser.parseSlice` will now ignore whitespace-only text nodes at the top of the slice.

prosemirror-view 0.14.2 (2016-11-30)

Bug fixes

FIX: Avoid unneeded selection resets which sometimes confused browsers.

prosemirror-view 0.14.2 (2016-11-29)

Bug fixes

Fix a bug where inverted selections weren't created in the DOM correctly.

prosemirror-view 0.14.1 (2016-11-29)

Bug fixes

Restores previously broken kludge that allows the cursor to appear after non-text content at the end of a line.

prosemirror-model 0.14.0 (2016-11-28)

New features

Parse rules now support `skip` (skip outer element, parse content) and `getContent` (compute content using custom code) properties.

The `DOMSerializer` class now exports a static `renderSpec` method that can help render DOM spec arrays.

prosemirror-state 0.14.0 (2016-11-28)

New features

Selection actions now have a `time` field and an (optional) `origin` field.

prosemirror-view 0.14.0 (2016-11-28)

Breaking changes

Wrapping decorations are now created using the `nodeName` property. The `wrapper` property is no longer supported.

The `onUnmountDOM` prop is no longer supported (use a node view with a `destroy` method instead).

The `domSerializer` prop is no longer supported. Use node views to configure editor-specific node representations.

New features

Widget decorations can now be given a `key` property to prevent unneccesary redraws.

The `EditorView` class now has a `destroy` method for cleaning up.

The `handleClickOn` prop and friends now receive a `direct` boolean argument that indicates whether the node was clicked directly.

Widget decorations now support a `stopEvent` option that can be used to control which DOM events that pass through them should be ignored by the editor view.

You can now specify custom node views for an editor view, which give you control over the way node of a given type are represented in the DOM. See the related RFC.

prosemirror-view 0.13.2 (2016-11-15)

Bug fixes

Fixes an issue where widget decorations in the middle of text nodes would sometimes disappear.

prosemirror-view 0.13.1 (2016-11-15)

Bug fixes

Fixes event handler crash (and subsequent bad default behavior) when pasting some types of external HTML into an editor.

prosemirror-model 0.13.0 (2016-11-11)

Breaking changes

`ResolvedPos.sameDepth` is now called `ResolvedPos.sharedDepth`, and takes a raw, unresolved position as argument.

New features

`DOMSerializer`'s `nodes` and `marks` properties are now public.

`ContentMatch.findWrapping` now takes a third argument, `marks`. There's a new method `findWrappingFor` that accepts a whole node.

Adds `Slice.maxOpen` static method to create maximally open slices.

DOM parser objects now have a `parseSlice` method which parses an HTML fragment into a `Slice`, rather than trying to create a whole document from it.

prosemirror-transform 0.13.0 (2016-11-11)

Bug fixes

Fix issue where `Transform.replace` would, in specific circumstances, unnecessarily drop content.

New features

The new `Transform` method `replaceRange`, `replaceRangeWith`, and `deleteRange` provide a way to replace and delete content in a 'do what I mean' way, automatically expanding the replaced region over empty parent nodes and including the parent nodes in the inserted content when appropriate.

prosemirror-state 0.13.0 (2016-11-11)

Breaking changes

`EditorTransform.replaceSelection` now takes a slice, no longer a node. The new `replaceSelectionWith` method should be used to replace the selection with a node. Until the next release, calling it the old way will still work and emit a warning.

Bug fixes

The documentation for `applyAction` now actually reflects the arguments this method is given.

New features

A state field's `applyAction` method is now passed the previous state as 4th argument, so that it has access to the new doc and selection.

`EditorTransform.replaceSelection` now accepts a slice (or, as before, as a node), and uses a revised algorithm, relying on the `defining` node flag.

The `TextSelection` and `NodeSelection` classes now have a static `create` convenience method for creating selections from unresolved positions.

Allow transform actions to be extended during dispatch using `extendTransformAction`. Introduce `sealed` flag to indicate when this is not safe.

A new utility function `NodeSelection.isSelectable` can be used to test whether a node can be the target of a node selection.

prosemirror-view 0.13.0 (2016-11-11)

Breaking changes

Selecting nodes on OS X is now done with cmd-leftclick rather than ctrl-leftclick.

Bug fixes

Pasting text into a code block will now insert the raw text.

Widget decorations at the start or end of a textblock no longer block horizontal cursor motion through them.

Widget nodes at the end of textblocks are now reliably drawn during display updates.

New features

`DecorationSet.map` now takes an options object which allows you to specify an `onRemove` callback to be notified when remapping drops decorations.

The `transformPastedHTML` and `transformPastedText` props were (re-)added, and can be used to clean up pasted content.

prosemirror-commands 0.13.0 (2016-11-11)

New features

The `autoJoin` function allows you to wrap command functions so that when the command makes nodes of a certain type occur next to each other, they are automatically joined.

prosemirror-view 0.12.2 (2016-11-02)

Bug fixes

Inline decorations that span across an empty textblock no longer crash the display drawing code.

prosemirror-view 0.12.1 (2016-11-01)

Bug fixes

Use a separate document to parse pasted HTML to better protect against cross-site scripting attacks.

Specifying multiple classes in a decoration now actually works.

Ignore empty inline decorations when building a decoration set.

prosemirror-history 0.12.1 (2016-11-01)

Bug fixes

Fix crash in undo or redo commands when the history is empty.

prosemirror-transform 0.12.1 (2016-11-01)

Bug fixes

Fix bug in `Transform.setBlockType` when used in a transform that already has steps.

prosemirror-model 0.12.0 (2016-10-21)

Breaking changes

Drops support for some undocumented options to the DOM serializer that were used by the view.

Bug fixes

When rendering DOM attributes, only ignore null values, not all falsy values.

prosemirror-transform 0.12.0 (2016-10-21)

Breaking changes

Mapped positions now count as deleted when the token to the side specified by the `assoc` parameter is deleted, rather than when both tokens around them are deleted. (This is usually what you already wanted anyway.)

prosemirror-state 0.12.0 (2016-10-21)

Breaking changes

The interace to `EditorState.toJSON` and `EditorState.fromJSON` has changed.

The way plugins declare their state field has changed. Only one state field per plugin is supported, and state fields no longer have hard-coded names. `Plugin.getState` is the way to access plugin state now.

Plugin dependencies are no longer supported.

`Plugin.reconfigure` is gone. Plugins are now always created with `new Plugin`.

Plugins no longer have a `config` field.

Bug fixes

Node selections are now properly dropped when mapped over a change that replaces their nodes.

New features

Plugin keys can now be used to find plugins by identity.

Transform actions now have a `time` field containing the timestamp when the change was made.

prosemirror-view 0.12.0 (2016-10-21)

Breaking changes

The return value of `EditorView.posAtCoords` changed to contain an `inside` property pointing at the innermost node that the coordinates are inside of. (Note that the docs for this method were wrong in the previous release.)

Bug fixes

Reduce reliance on shift-state tracking to minimize damage when it gets out of sync.

Fix bug that'd produce bogus document positions for DOM positions inside non-document nodes.

Don't treat fast ctrl-clicks as double or triple clicks.

New features

Implement decorations, a way to influence the way the document is drawn. Add the `decorations` prop to specify them.

prosemirror-keymap 0.12.0 (2016-10-21)

Breaking changes

Key names are now based on `KeyboardEvent.key` instead of `.code`. This means that, for character-producing keys, you'll want to use the character typed, not the key name. So `Ctrl-Z` now means uppercase Z, and you'll usually want `Ctrl-z` instead. Single-quoted key names are no longer supported.

prosemirror-history 0.12.0 (2016-10-21)

Breaking changes

The `history` export is now a function that creates a history plugin, rather than a plugin instance.

New features

Add a `newGroupDelay` plugin option. This brings back the behavior where pausing between edits will automatically cause the history to put subsequent changes in a new undo event.

prosemirror-commands 0.12.0 (2016-10-21)

Bug fixes

Fix crash when backspacing into nodes with complex content expressions.

prosemirror-schema-basic 0.12.0 (2016-10-21)

Bug fixes

Don't treat <b style=font-weight: normal> as strong when parsing. (Google Docs puts such ridiculous HTML on the clipboard.)

prosemirror-view 0.11.2 (2016-10-04)

Bug fixes

Pass actual event object to `handleDOMEvent`, rather than just its name.

Fix display corruption caused by using the wrong state as previous version during IME.

prosemirror-model 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

The JSON representation of marks has changed from `{"_": "type", "attr1": "value"}` to `{"type": "type", "attrs": {"attr1": "value"}}`, where `attrs` may be omitted when the mark has no attributes.

Mark-related JSON methods now live on the `Mark` class.

The way node and mark types in a schema are defined was changed from defining subclasses to passing plain objects (`NodeSpec` and `MarkSpec`).

DOM serialization and parsing logic is now done through dedicated objects (`DOMSerializer` and `DOMParser`), rather than through the schema. It is now possible to define alternative parsing and serializing strategies without touching the schema.

New features

The `Slice` class now has an `eq` method.

The `Node.marksAt` method got a second parameter to indicate you're interested in the marks *after* the position.

prosemirror-transform 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

The `Remapping` class was renamed to `Mapping` and works differently (simpler, grows in only one direction, and has provision for mapping through only a part of it).

`Transform` objects now build up a `Mapping` instead of an array of maps.

`PosMap` was renamed to `StepMap` to make it clearer that this applies only to a single step (as opposed to `Mapping`.

The arguments to `canSplit` and `split` were changed to make it possible to specify multiple split-off node types for splits with a depth greater than 1.

Rename `joinable` to `canJoin`.

New features

Steps can now be merged in some circumstances, which can be useful when storing a lot of them.

prosemirror-state 0.11.0 (2016-09-21)

Breaking changes

New module inheriting the `Selection` and `EditorTransform` abstraction, along with the persistent state value that is now separate from the display logic, and the plugin system.

`Selection.findAtStart/End` was renamed to `Selection.atStart/End`, and `Selection.findNear` to `Selection.near`.

prosemirror-view 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module from the old `edit` submodule. Completely new approach to managing the editor's DOM representation and input.

Event handlers and options are now replaced by props. The view's state is now 'shallow', represented entirely by a set of props, one of which holds an editor state value from the state module.

When the user interacts with the editor, it will pass an action to its `onAction` prop, which is responsible for triggering an view update.

The `markRange` system was dropped, to be replaced in the next release by a 'decoration' system.

There is no keymap support in the view module anymore. Use a keymap plugin for that.

The undo history is now a separate plugin.

CSS needed by the editor is no longer injected implicitly into the page. Instead, you should arrange for the `style/prosemirror.css` file to be loaded into your page.

New features

The DOM parser and serializer used to interact with the visible DOM and the clipboard can now be customized through props.

You can now provide a catch-all DOM event handler to get a first chance at handling DOM events.

The `onUnmountDOM` can be used to be notified when a piece of the document DOM is thrown away (in case cleanup is needed).

prosemirror-keymap 0.11.0 (2016-09-21)

Breaking changes

New module, takes the same role as the old built-in keymap support in the `ProseMirror` class.

prosemirror-inputrules 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

You can now add this plugin multiple times to add different sets of rules to an editor (if you want). It is not possible to change the set of rules of an existing plugin instance.

Rules no longer take a `filter` argument.

The signature of the `handler` callback for a rule changed.

prosemirror-history 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module. Now acts as a plugin that can be omitted or replaced by a different implementation if desired.

Merging subsequent changes into a single undo 'event' is now done by proximity in the document (the changes must touch) rather than in time. This will probably have to be further refined.

prosemirror-collab 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

Interface adjusted to work with the new plugin system.

New features

When receiving changes, the module now generates a regular transform action instead of hard-setting the editor's document. This solves problematic corner cases for code keeping track of the document by listening to transform actions.

prosemirror-commands 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

The interface for command functions was changed to work with the new state/action abstractions.

prosemirror-schema-basic 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

No longer exports the specs for the nodes and marks separately, since they are now plain objects, not subclasses. They are still exported through nodes and marks objects.

The list-related nodes were moved to the schema-list module.

prosemirror-schema-list 0.11.0 (2016-09-21)

Breaking changes

New module combining the node specs from schema-basic, and the list-related commands from the commands module.

prosemirror-schema-table 0.11.0 (2016-09-21)

Breaking changes

Moved into a separate module.

Node types adjusted to the new node spec interface.

Commands adjusted to the new command interface.

0.10.1 (2016-09-12)

Bug fixes

Fixes a problem where the DOM parser would produce corrupted documents in some corner cases.

Makes the editor useable on IE again by not assuming `document` has a `contains` method.

0.10.0 (2016-08-26)

Bug fixes

Fixed several issues in the handling of clicks on and near leaf nodes.

Fix bug where `liftTarget` would produce false positives.

Improve support for using ProseMirror in a shadow DOM.

New features

The `ProseMirror.on.domPaste` event can now be used to entirely override the handling of paste events.

The new `ProseMirror.root` property returns the document or shadow DOM root that the editor is part of.

New method `PosMap.forEach` to easily iterate over the changed ranges in a position map.

Marked ranges now support an `elementBefore` option to insert a DOM node before the range.

0.9.1 (2016-07-29)

Bug fixes

Fix DOM node leak in the creation of menu SVG icons.

Fix regression where `clickOn` and `doubleClickOn` handlers weren't called on clicked leaf nodes.

Fix crash in DOM parser when parsing a DOM change.

0.9.0 (2016-07-26)

Bug fixes

Pasting a single leaf node (a node that doesn't allow content) works again.

Parsing Markdown text with code blocks (indented or fenced) works again.

Fixes a bug where clicking a selectable leaf node would create a selection before that node.

Fix the way `requestAnimationFrame` is called to please Internet Explorer.

New features

Mouse-related event handlers (`click`, `clickOn`, `doubleClick`, `doubleClickOn`, `contextMenu`) are now passed the original DOM event.

Adds a new module `schema-tables`, that exports node types and commands for tables, along with a way to easily add them to your schema.

0.8.3 (2016-06-28)

Bug fixes

When pasting text, assign some basic structure to it by inserting `<p>` and `<br>` tags.

Fix a bug in `EditorTranform.selection`.

Disable the `setBlockType` command for non-textblock blocks.

Make `splitBlock` handle content restrictions better.

0.8.2 (2016-06-21)

Bug fixes

Fix `OrderedMap.append` and `OrderedMap.prepend` to allow raw-object arguments.

Fix bug where `Node.textContent` didn't actually return a value.

Make the `wrapInList` command more robust in the face of schema constraints.

0.8.1 (2016-06-16)

Bug fixes

Fixes `addActiveMark` and `removeActiveMark` ignoring marks from the document, and thus accidentally resetting the set of marks when used.

Properly export the parser, serializer, and state classes from the `markdown` module.

0.8.0 (2016-06-15)

Breaking changes

The `src/` directory no longer uses ES6 modules, and is now CommonJS-based instead. With ES6 support—*except* for modules—being pretty much complete in browsers and node, that was the only thing holding us back from running the code directly, without compilation.

There is no longer a default schema in the `model` module. The new `schema-basic` module exports the schema, node types, and mark types that used to make up the default schema.

The `schema` option is no longer optional (though it is implied if you pass a `doc` option).

The `Command` abstraction was entirely removed. When the docs talk about *commands* now, they refer to plain functions that take an editor instance as argument and return a boolean indicating whether they could perform their action.

Keymaps now map keys to such command functions. The basic keymap for an editor is no longer inferred from commands, but is now determined by the `keymap` option. The default value contains a minimal set of bindings not related to any schema elements.

Defining new options is no longer a thing. Modules that used to use `defineOption` now export plugins.

Changing options at run-time through `setOption` is no longer supported, since the remaining built-in options don't require this.

The `docFormat` option, `getContent`/`setContent` editor methods, and the `format` module have been dropped. You need to explicitly convert between formats yourself now.

The DOM parsing and serializing from the `format` module was moved into the `model` module, as the `toDOM` and `parseDOM` methods.

The conversions to and from text and HTML were dropped. HTML can be done by going through the DOM representation, the text format conversions were never very well defined to begin with.

Declaring the way a node or mark type is parsed or serialized was simplified. See the new `toDOM`, `matchDOMTag` and `matchDOMStyle` properties.

The `SchemaItem` class, along with the `register` and `updateAttrs` static methods on that class, have been removed. `NodeType` and `MarkType` no longer share a superclass, and registering values on such types is no longer a thing.

`Textblock` is no longer a class that you derive node type classes from. Just use `Block`, and the `isTextblock` getter will return the right value based on the node type's content.

Event handlers are no longer registered with node-style `on` methods, but attached to specific values representing an event source. The subscription module is used for this. The event emitters for an editor instance are grouped in its `on` property. So now you'd say `pm.on.change.add(...)` to register a change handler.

The event-like node type methods `handleClick`, `handleDoubleClick`, and `handleContextMenu` are no longer supported. Instead, you're expected to use the `click`, `clickOn`, `doubleClick`, `doubleClickOn`, and `contextMenu` event emitters.

The `removed` event on marked ranges was replaced by an `onRemove` option.

The `apply` method on an editor now always returns the transform, whereas it used to return `false` if no change was made. Its `scroll` property (containing a commonly used options object) was removed.

The `checkPos` method on editor objects was removed.

The `rank` argument to `addKeymap` was renamed `priority` and its meaning was inverted.

The `setMark` method on editor instances was removed. Its role is mostly taken over by the `toggleMark` command.

The functionality from the `ui/update` module was moved into the `edit` module and is now available through the `scheduleDOMUpdate`, `unscheduleDOMUpdate`, and `updateScheduler` methods.

Selection objects now contain resolved positions, because that's what you need 99% of the time when you access them. Their old properties are still there, in addition to `$from`, `$to`, `$anchor`, and `$head` properties. The constructors of the selection classes expect resolved positions now.

The `findDiffStart` and `findDiffEnd` functions were moved to methods on `Fragment`.

Some transformation methods are now less vague in their parameters. `wrap` requires the range to wrap and the entire set of wrappers as argument, `lift` expects a range and target depth.

The `findWrapping` and `liftTarget` functions are used to compute these before trying to apply the transformation. They replace `canWrap` and `canLift`.

The structure of the markdown parser and serializer was changed to no longer rely on `SchemaItem.register`. Adjusting the parser and serializer now works differently (you explicitly create an object rather than relying on information attached to node types).

The `autoinput` module was removed. The `inputrules` module now exports a plugin that can be used to add input rules, along with a number of basic input rules and helper functions to create schema-specific input rules.

The `menu` module is now a single module, which exports the menu primitives along with the `menuBar` and `tooltipMenu` plugins.

Menu construction is no longer entangled with command definitions. The `MenuCommand` class was replaced with a `MenuItem` class, which directly declares the things it would previously get from a command. The concept of a `MenuGroup` was dropped (we just use arrays, since they are always static). Some helper functions for creating menu items, along with some basic icons, are exported from the `menu` module.

The `ui` module is now a single module, which exports the `Tooltip` class and the prompt-related functionality. Now that command parameters no longer exist, the interface for creating a prompt was also changed.

The events emitted by the `collab` module (which now exports a plugin) are now subscriptions on the plugin state object, named `mustSend` and `receivedTransform`.

`Step.register` was renamed to `Step.jsonID`.

All non-essential CSS rules were removed from the core.

Bug fixes

Several issues where the code didn't properly enforce the content constraints introduced in 0.7.0 were fixed.

When pasting content into an empty textblock, the parent node it was originally copied from is now restored, when possible.

Several improvements in the handling of composition and input events (mostly used on mobile platforms). Fixes problem where you'd get a strange selection after a complex composition event.

Make by-word deletion work properly on astral plane characters.

Fix leaked spacer node when the menu bar was disabled while it was floating.

New features

Plugins are objects used to attach (and detach) a specific piece of functionality to an editor. Modules that extend the editor now export values of this type. The `plugins` option is the easiest way to enable plugins.

The `contextAtCoords` method provides more precise information about a given position (including the exact nodes around the position) than the existing `posAtCoords` method.

The `EditorTransform` abstraction is now more closely integrated with the editor selection, and you can call `setSelection` on it to update the selection during a transform.

The new `applyAndScroll` method on editor transforms provides a convenient shortcut for the common case of applying a transformation and scrolling the selection into view.

The new methods `addActiveMark` and `removeActiveMark` provide explicit control over the active stored marks.

The `edit` module now exports an object `commands` containing a number of command functions and functions that produce command functions. Most of the old command objects have an equivalent here.

The `forEach` method on nodes and fragments now also passes the node's index to its callback.

Nodes now have a `textBetween` method that retrieves the text between two positions.

A new `NodeRange` abstraction (created with the `blockRange` method on positions) is used to specify the range that some of the transformation methods act on.

Node types got two new variants of their `create` method: `createChecked`, which raises an error if the given content isn't valid (full) content for the node, and `createAndFill`, which will automatically insert required nodes to make the content valid.

The `fixContent` method on node types was removed.

You can now pass a second argument to the `Schema` constructor, and access its value under the schema object's `data` property, to store arbitrary user data in a schema.

The transform module now exports an `insertPoint` function for finding the position at which a node can be inserted.

The `OrderedMap` class received a new method, `addBefore`.

A new module, `example-setup` provides a plugin that makes it easy to set up a simple editor with the basic schema and the key bindings, menu items, and input rules that used to be the default.

The list node types in the basic schema now require the first child of a list item to be a regular paragraph. The list-related commands are now aware of this restriction.

0.7.0 (2016-05-19)

Breaking changes

The following properties on node types have lost their meaning: `kind`, `contains`, `canBeEmpty` and `containsMarks`. The `NodeKind` type is also gone.

The information they used to encode is now put in a content expression, which is part of the schema spec, not the node. Such expressions can refer directly to other nodes in the schema (by name).

`SchemaSpec` is now an interface, not a class. Its `nodes` field refers to `NodeSpec` objects, rather than directly to `NodeType` constructors. These hold not only a constructor but also a content expression and optionally a group identifier.

The `NodeType` methods `canContain`, `canContainFragment`, `canContainMark`, `canContainContent`, and `canContainType` are gone, since they can't accurately express the constraints of the new content expressions.

Instead, nodes now expose `canReplace`, `canReplaceWith`, and `canAppend`. The `contentMatchAt` method gets you a `ContentMatch` object which provides further ways to reason about content.

`NodeType.findConnection` is now at `ContentMatch.findWrapping`, and takes and returns attributes as well as node types.

Mark types lost their `rank` property, as their ordering is now determined by the order in which they appear in the schema spec.

Transform steps are now regular classes, `AddMarkStep`, `RemoveMarkStep`, `ReplaceStep`, and `ReplaceAroundStep`. `Transform.step` now only takes a step object, not separate values. The `"join"`, `"split"`, and `"ancestor"` step types have been superseded by `ReplaceStep` and `ReplaceAroundStep`.

The collaborative editing protocol was changed, to resolve a synchronization problem. See the guide for an overview of the new protocol.

New features

Node nesting is now expressed through a more powerful mechanism, content expressions.

The `ContentMatch` class provides a way to apply and reason about such content expressions.

The new `OrderedMap` class makes it possible to extend and modify the sets of nodes and marks in a schema while keeping control over their order.

Since splitting isn't always possible any more, a new function `canSplit` is exported by the `transform` module.

The new options `scrollTreshold` and `scrollMargin` provide more control over scrolling behavior.

`nodesBetween` now passes the node's index to its callback as fourth argument.

Node types gained a getter `isLeaf` to conveniently test whether they allow content.

Resolved positions got a new method `indexAfter`, and their methods that expect a depth allow the argument to be omitted to specify the position's own depth, or a negative integer to be passed to specify a depth relative to the position's depth.

0.6.1 (2016-04-15)

Bug fixes

Composition (IME) input is now more robust. This mostly effects Android browsers, where typing is now less buggy.

The iOS virtual keyboard's default case should now update as you type (rather than being stuck in whatever state it was in when you started typing).

Text input read through composition or input events now fires input rules.

A problem where transform filters could corrupt the undo history has been fixed.

0.6.0 (2016-04-13)

Breaking changes

Positions in the document are now represented by integers, rather than `Pos` objects. This means that *every* function parameter, return value, or property that used to be a `Pos` is now a number instead.

Be *extra* wary about functions that return an optional position—0 is a valid position now, so if your code is just checking `if (pos) ...`, it'll break when getting a 0.

The `countCoordsAsChild`, `handleClick`, `handleDoubleClick`, and `handleContextMenu` methods on node types, which used to take a path as an array of numbers, now get a single number pointing at the node's position in the document instead.

The `"selectNodeLeft/Right/Up/Down"` commands, which were a hack to make node selection work, are now no longer exposed as commands.

The key bindings for block types changed again, due to the old ones still clashing with default OS X bindings. They are now prefixed with Shift-Ctrl (rather than Shift-Cmd on OS X).

Nodes lost the `size` and `width` properties, and now expose a `nodeSize` property instead, which is the total size of the node. The `size` attribute on fragments changed meaning to point at the total size of the fragment's children (rather than their count).

Node iterators are gone, and replaced by index-based access using the `childCount` property and the `child` and `maybeChild` accessors.

The `chunkBefore` and `chunkAfter` methods on nodes are replaced by a `childBefore` and `childAfter` method with the same role but slightly different semantics.

`Node.slice` now returns a `Slice`. `Node.sliceBetween` is gone. The method that just returns a reduced `Node` is now called `cut` (and also present on fragments).

The node and fragment methods `splice`, `append`, `close`, `replaceDeep`, and the old `replace` are gone. Document manipulation is now best done in one shot using the new `replace` method, which replaces a range of the document with a `Slice`.

Since we are no longer using arrays of numbers to find nodes, `Node.path` is gone. To find out what an integer position points at, use `Node.resolve`, and then inspect the resulting `ResolvedPos` object.

`Node.nodeAfter` is now called `Node.nodeAt`. It does mostly the same thing, except that it now takes a number position.

`Node.nodesBetween` passes a start position for the current node, rather than mutable path, to its callback. `Node.inlineNodesBetween` is gone, since it is now very easy to do something like that with `nodesBetween`. `Node.descendants` is a new shorthand that iterates over *all* descendant nodes.

Fragments lost their `toArray`, `map`, and `some` methods, and otherwise mostly mirror the changes in the `Node` type.

The constant empty fragment now lives under `Fragment.empty` rather than `emptyFragment`.

Steps lost their `pos` property. They now only store a `from` and `to` (as numbers rather than `Pos` objects).

The result of applying a step no longer contains a position map. Those can be derived from a step without applying it now (using the `posMap` method). A failing step no longer returns `null`. Rather, a step result contains *either* an error message *or* an updated document.

You no longer need to provide a position map when inverting a step.

The `Mappable` interface's `map` method now returns a plain position, instead of a `MapResult`. Use the `mapResult` method if you need the additional information.

Position maps have gotten much simpler, and are created differently now.

Transforms no longer silently ignore failing steps unless you explicitly tell them to by using the `maybeStep` method. The `step` method, along with most of the other transformation methods, will raise an error when they can't be applied.

`Transform.replace` now takes a `Slice` object, rather than a full replacement document with start and end positions.

Bug fixes

An unsoundness in the collaborative editing algorithm's handling of replace steps has been fixed.

The SVG icons now also work when you have a `<base>` tag on your page.

Fix select-all on Firefox.

Fix crash in history compression.

Properly handle HTML sublists not wrapped in an `<li>` tag.

Prevent Ctrl-Enter and Ctrl-Backspace on OS X from messing up our document.

Handle the case where a `clipboardData` object is present but doesn't actually work (iOS).

New features

`ProseMirror.flush` now return a boolean indicating whether it redrew the display.

New data type, `Slice`, which represents a piece of document along with information about the nodes on both sides that are 'open' (can be joined to adjacent nodes when inserting it into a document).

The new `"transformPasted"` event can be used to transform pasted or dragged content, as a parsed `Slice`.

The `Node.eq` predicate can now be used to determine whether two nodes are equal.

Mark types can now control whether they are applied to text typed after such a mark with their `inclusiveRight` property.

The `join` and `lift` transform methods now have a `silent` parameter to suppress exceptions when they can not be applied.

The `type` parameter to `setNodeType` now defaults to the node's current type.

`toDOM`, `toHTML`, and `toText` now accept `Fragment` objects as well as nodes.

List items now have `lift` and `sink` commands.

0.5.1 (2016-03-23)

Bug fixes

Fix malformed call that caused any nodes rendered with `contenteditable=false` to be replaced by a bogus `<div>`.

0.5.0 (2016-03-22)

Bug fixes

ProseMirror now ignores most evens when not focused, so you can have focusable fields inside the editor.

The Markdown serializer is now a lot more clever about serializing mixed inline styles.

Event handlers unregistering themselves is now safe (used to skip next event handler).

New features

The default command parameter prompt UI now shows the command label and a submit button.

When joining an empty textblock with a non-empty one, the resulting block now gets the type of the non-empty one.

Node types can now handle double clicks with a `handleDoubleClick` method.

Undo and redo now restore the selection that was current when the history event was created.

The collab module now fires a `"collabTransform"` event when receiving changes.

The `"filterTransform"` event can now be used to cancel transforms.

Node kinds can now specify both their super- and sub-kinds.

0.4.0 (2016-02-24)

Breaking changes

The way valid parent-child relations for node types are specified was changed. Instead of relying on strings, node kinds are now objects that specify arbitrary sub-kind relations. The static `kinds` property on node types replaced by a non-static `kind` property holding such a kind object, and the `contains` property is now expected to hold a kind object instead of a string.

The keybindings for make-paragraph and make-heading were changed. To make the current textblock a paragraph, you now press Ctrl/Cmd-0, and to make it a heading, you press Ctrl/Cmd-N, where N is the level of the heading.

Bug fixes

Copy-pasting between ProseMirror instances is now more robust (the question of whether the selection cuts through nodes at the start and end of the fragment is preserved).

Selection management on mobile platforms should now be improved (no longer unusable, probably still quite buggy).

Fix a problem where reading a change from the DOM was confused by trailing whitespace in a text block.

Fix a bug in scrolling things into view that would break scrolling of anything except the whole document.

Don't render menu dropdown elements unless they actuall have content.

Fix bug that would reset the selection when a `selectionChange` event handler tried to access it.

The selection classes are now properly exported from the `edit` module.

New features

Drop events can now be intercepted.

The `beforeTransform` event is now fired before a transform is applied.

Menu command icon specs can now provide a `dom` property to provide a piece of DOM structure as their icon.

DOM parser specs can now include a `selector` property to only run the parser on nodes that match the selector.

0.3.0 (2016-02-04)

Breaking changes

The way menu items for menu modules are configured now works differently, expecting types from the `menu` module. The way commands declare themselves to be part of a menu group is also different—the information previously stored in the `menuGroup` and `display` properties now goes, in a somewhat different format, in the `menu` property.

The command parameter prompting functionality was changed. The `paramHandler` option has been replaced by a `commandParamPrompt` option. The prompting functionality now lives in the `prompt` module, and should be easier to extend.

The styling and animation of menus and tooltips was changed to be simpler and easier to maintain. Fancy UI looks are now considered out of scope for this module, and something that should be implemented in third-party modules.

Bug fixes

Selection on mobile should work much better now (though probably still far from perfect).

Pressing enter on a mobile device will no longer corrupt the display.

New features

New menu building blocks in the `menu` module allow more control and flexibility when defining menus.

`ProseMirror.history` is now documented and received a new `isAtVersion` method to check whether an editor is 'clean' relative to a given version.

0.2.0 (2016-01-28)

Breaking changes

- The `register` method's signature changed, requiring an item name as well as a namespace. Most uses of the schema registry now use that name to replace a field that was previously part of the registered value. For example, command specs no longer have a `name` field, but use the registry name. (This was needed to make it possible to selectively override or disable registered values in classes that derive from schema items.)
- `InputRule`s no longer have a `name` field, and the corresponding constructor parameter was removed. `removeInputRule` now takes a rule object rather than a name string.
- Items in the 'insert' and node type menus are now added with `register` (under `"insertMenu"` and `"textblockMenu"`) rather then with a direct property.
- The JSON representation of marks changed. This release will still parse the old representation (spitting out a warning). The next release won't, so if you're storing JSON data make sure you parse and re-save at least once with 0.2.0 before upgrading further.
- The function passed to the `UpdateScheduler` constructor now starts in the DOM write phase (used to be read).
- The `"flushed"` event was removed.
- The `selectedDoc` and `selectedText` methods were removed.

Bug fixes

- The Markdown parser now throws an error when encountering a token it doesn't know how to handle.
- The menubar will no longer hide the top of the content when the controls inside of it line-wrap.
- Dropped content is now properly selected.

- Less fragile rules for curly quote autocompletion.
- The DOM parser now ignore non-displaying tags (like `<script>` and `<style>`).
- Our `package.json` now has a `"main"` field.
- Fix bug where trailing newlines in code blocks would not be visible.
- Fix several issues with locating positions in the DOM that occurred for node types that wrapped their content in more than a single element (such as the default code blocks).

New features

- The menu/menu module now exposes an object `paramTypes` which allows you to add or redefine the types of parameters that can be rendered.
- The ui/update module now exports `scheduleDOMUpdate` and `unscheduleDOMUpdate` functions to schedule synchronized DOM updates.
- Schema items now expose a `cleanNamespace` method to 'forget' values registered on superclasses.
- The computation of registered values on schema items can now be delayed to schema-instantiation-time with the `registerComputed` method.
- Schema items can now register `"configureMarkdown"` items to influence the way the parser library is initialized.
- The `"splitBlock"` command will now split off a plain paragraph when executed at the start of a different kind of textblock.
- Node types may now define a `handleContextMenu` method to intercept context menu events inside of them.
- The `Heading` node type now supports a `maxLevel` property that subclasses can use to configure the maximum heading level.
- Node types can now declare themselves to be `draggable`.
- Node selections can now be dragged.

0.1.1 (2016-01-11)

Initial release.