In-place editing with Twitter Bootstrap, jQuery UI or pure jQuery

vitalets.github.io/x-editable/docs.html

Getting started

1. Decide which core library you want to use:
   - **Bootstrap**
   - **jQuery UI**
   - **only jQuery (+ Poshytip)**

   Include it on your page. Examples below are for *bootstrap*:

   ```
   <link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css"
   rel="stylesheet">
   <script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
   <script src="//netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js"></script>
   ```

2. Download corresponding X-editable build and include it on your page:

   ```
   <link href="bootstrap-editable/css/bootstrap-editable.css" rel="stylesheet">
   <script src="bootstrap-editable/js/bootstrap-editable.js"></script>
   ```

   Note to include x-editable after core library (bootstrap, jquery-ui)!

3. Markup elements that should be editable. Usually it is `<A>` element with additional `data-*` attributes

   ```
   <a href="#" id="username" data-type="text" data-pk="1" data-url="/post" data-title="Enter
   username">superuser</a>
   ```

   Main attributes you should define are:
   - `type` - type of input (text, textarea, select, etc)
   - `url` - url to server-side script to process submitted value (`/post` , `post.php` etc)
   - `pk` - primary key of record to be updated (ID in db)
   - `id` or `name` - name of field to be updated (column in db). Taken from `id` or `data-name` attribute
   - `value` - initial value. Usefull for select, where value is integer key of text to be shown. If *empty* - will be taken from element html contents

4. Setup editable mode: **inline** or **popup** (default)

   ```
   //turn to inline mode
   $.fn.editable.defaults.mode = 'inline';
   ```

5. Apply **editable()** method to these elements

   ```
   $(document).ready(function() {
       $('#username').editable();
   });
   ```

   Alternatively, you can set all options via javascript

   ```
   <a href="#" id="username">superuser</a>
   ```

```
$('#username').editable({
    type: 'text',
    pk: 1,
    url: '/post',
    title: 'Enter username'
});
```

6. Frontend ready!
   View starter template Download starter zip

   Open your page and click on element. Enter new value and submit form. It will send ajax request with new value to `/post`.
   Request contains `name`, `value` and `pk` of record to be updated:

```
POST /post
{
    name:  'username',  //name of field (column in db)
    pk:    1            //primary key (record id)
    value: 'superuser!' //new value
}
```

7. Write backend part:
   X-editable has no limitation to server-side part: you can write it on any language you prefer.
   For example, you want to validate submitted value on server:
   - If value is valid, you should return **HTTP status 200 OK**. Element on page will be updated automatically. No response body required.
   - If value is not valid, you should return **HTTP status != 200** (e.g. *400 Bad request*) with error message in response body. Element on page will not be updated and editable form will display error message.

   Default request method is **POST**, you can change it via *defaults* config:

```
$.fn.editable.defaults.ajaxOptions = {type: "PUT"};
```

   JSON response:
   If your server returns JSON, you can always send HTTP status 200 with error flag in response body.
   To process it use `success` handler:

```
//assume server response: 200 Ok {status: 'error', msg: 'field cannot be empty!'}

$('#username').editable({
    ...
    success: function(response, newValue) {
        if(response.status == 'error') return response.msg; //msg will be shown in editable
form
    }
});
```

   Work LOCALLY:
   If you don't want to send value on server, just keep empty `url` option. You can process value in `success` handler:

```
$('#username').editable({
    type: 'text',
    title: 'Enter username',
    success: function(response, newValue) {
        userModel.set('username', newValue); //update backbone model
    }
});
```

$().editable(options)

Makes editable any HTML element on the page. Applied as jQuery method.

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.
Please see also input dependent options.
Additionally, you can use core options of popup from their docs: bootstrap-popover, jquery-ui-tooltip.

Name	Type	Default	Description
ajaxOptions since 1.1.1	object	null	Additional options for submit ajax request. List of values: http://api.jquery.com/jQuery.ajax `ajaxOptions: {` `  type: 'put',` `  dataType: 'json'` `}`
anim	string	false	Animation speed (inline mode only)
autotext	string	'auto'	Allows to automatically set element's text based on it's value. Can be `auto|always|never`. Useful for select and date. For example, if dropdown list is `{1: 'a', 2: 'b'}` and element's value set to `1`, it's html will be automatically set to `'a'`. `auto` - text will be automatically set only if element is empty. `always|never` - always(never) try to set element's text.
defaultValue since 1.4.6	string|object	null	Value that will be displayed in input if original field value is empty (`null|undefined|''`).
disabled	boolean	false	Sets disabled state of editable
display since 1.2.0	function|boolean	null	Callback to perform custom displaying of value in element's text. If `null`, default input's display used. If `false`, no displaying methods will be called, element's text will never change. Runs under element's scope. ***Parameters:*** • `value` current value to be displayed • `response` server response (if display called after ajax submit), since 1.4.0 For *inputs with source* (select, checklist) parameters are different: • `value` current value to be displayed • `sourceData` array of items for current input (e.g. dropdown items) • `response` server response (if display called after ajax submit), since 1.4.0 To get currently selected items use `$.fn.editableutils.itemsByValue(value, sourceData)`. `display: function(value, sourceData) {` `  //display checklist as comma-separated values` `  var html = [],` `  checked = $.fn.editableutils.itemsByValue(value, sourceData);` `  if(checked.length) {` `  $.each(checked, function(i, v) { html.push($.fn.editableutils.escape(v.text)); });` `  $(this).html(html.join(', '));` `  } else {` `  $(this).empty();` `  }` `}`
emptyclass since 1.4.1	string	editable-empty	Css class applied when editable text is empty.

Name	Type	Default	Description
emptytext	string	'Empty'	Text shown when element is empty.
error since 1.4.4	function	null	Error callback. Called when request failed (response status != 200). Usefull when you want to parse error response and display a custom message. Must return **string** - the message to be displayed in the error block. `error: function(response, newValue) {` `  if(response.status === 500) {` `  return 'Service unavailable. Please try later.';` `  } else {` `  return response.responseText;` `  }` `}`
highlight since 1.4.5	string|boolean	#FFFF80	Color used to highlight element after update. Implemented via CSS3 transition, works in modern browsers.
mode since 1.4.0	string	'popup'	Mode of editable, can be `popup` or `inline`
name	string	null	Name of field. Will be submitted on server. Can be taken from `id` attribute
onblur since 1.1.1	string	'cancel'	Action when user clicks outside the container. Can be `cancel|submit|ignore`. Setting `ignore` allows to have several containers open.
params	object|function	null	Additional params for submit. If defined as `object` - it is **appended** to original ajax data (pk, name and value). If defined as `function` - returned object **overwrites** original ajax data. `params: function(params) {` `  //originally params contain pk, name and value` `  params.a = 1;` `  return params;` `}`
pk	string|object|function	null	Primary key of editable object (e.g. record id in database). For composite keys use object, e.g. `{id: 1, lang: 'en'}`. Can be calculated dynamically via function.
placement	string	'top'	Placement of container relative to element. Can be `top|right|bottom|left`. Not used for inline container.
savenochange since 1.2.0	boolean	false	Whether to save or cancel value when it was not changed but form was submitted

Name	Type	Default	Description
selector since 1.4.1	string	null	If selector is provided, editable will be delegated to the specified targets. Usefull for dynamically generated DOM elements. **Please note**, that delegated targets can't be initialized with `emptytext` and `autotext` options, as they actually become editable only after first click. You should manually set class `editable-click` to these elements. Also, if element originally empty you should add class `editable-empty`, set `data-value=""` and write emptytext into element: `<div id="user">` `  <!-- empty -->` `  <a href="#" data-name="username" data-type="text" class="editable-click editable-empty" data-value="" title="Username">Empty</a>` `  <!-- non-empty -->` `  <a href="#" data-name="group" data-type="select" data-source="/groups" data-value="1" class="editable-click" title="Group">Operator</a>` `</div>` `<script>` `$('#user').editable({` `  selector: 'a',` `  url: '/post',` `  pk: 1` `});` `</script>`
send	string	'auto'	Strategy for sending data on server. Can be `auto|always|never`. When 'auto' data will be sent on server **only if pk and url defined**, otherwise new value will be stored locally.
showbuttons since 1.1.1	boolean|string	true	Where to show buttons: left(true)|bottom|false Form without buttons is auto-submitted.
success	function	null	Success callback. Called when value successfully sent on server and **response status = 200**. Usefull to work with json response. For example, if your backend response can be `{success: true}` or `{success: false, msg: "server error"}` you can check it inside this callback. If it returns **string** - means error occured and string is shown as error message. If it returns **object like** `{newValue: <something>}` - it overwrites value, submitted by user. Otherwise newValue simply rendered into element. `success: function(response, newValue) {` `  if(!response.success) return response.msg;` `}`
toggle	string	'click'	How to toggle editable. Can be `click|dblclick|mouseenter|manual`. When set to `manual` you should manually call `show/hide` methods of editable. **Note**: if you call `show` or `toggle` inside **click** handler of some DOM element, you need to apply `e.stopPropagation()` because containers are being closed on any click on document. `$('#edit-button').click(function(e) {` `  e.stopPropagation();` `  $('#username').editable('toggle');` `});`
type	string	'text'	Type of input. Can be `text|textarea|select|date|checklist` and more
unsavedclass since 1.4.1	string	editable-unsaved	Css class applied when value was stored but not sent to server (`pk` is empty or `send = 'never'`). You may set it to `null` if you work with editables locally and submit them together.

Name	Type	Default	Description
url	string|function	null	Url for submit, e.g. `'/post'` If function - it will be called instead of ajax. Function should return deferred object to run fail/done callbacks. `url: function(params) {` `  var d = new $.Deferred();` `  if(params.value === 'abc') {` `  return d.reject('error message'); //returning error via deferred object` `  } else {` `  //async saving data in js model` `  someModel.asyncSaveMethod({` `  ...,` `  success: function(){` `  d.resolve();` `  }` `  });` `  return d.promise();` `  }` `}`
validate	function	null	Function for client-side validation. If returns string - means validation not passed and string showed as error. Since 1.5.1 you can modify submitted value by returning object from `validate` : `{newValue: '...'}` or `{newValue: '...', msg: '...'}` `validate: function(value) {` `  if($.trim(value) == '') {` `  return 'This field is required';` `  }` `}`
value	mixed	element's text	Initial value of input. If not set, taken from element's text. Note, that if element's text is empty - text is automatically generated from value and can be customized (see `autotext` option). For example, to display currency sign: `<a id="price" data-type="text" data-value="100"></a>` `<script>` `$('#price').editable({` `  ...` `  display: function(value) {` `  $(this).text(value + '$');` `  }` `})` `</script>`

All methods can be called as `$().editable('method', params);`

Method	Parameters	Description
$().editable(options)	**options** *Object*	jQuery method to initialize editable element. `$('#username').editable({` `  type: 'text',` `  url: '/post',` `  pk: 1` `});`
activate()	*none*	Activates input of visible container (e.g. set focus)
destroy()	*none*	Removes editable feature from element
disable()	*none*	Disables editable
enable()	*none*	Enables editable

Method	Parameters	Description
getValue()	**isSingle** *Bool* whether to return just value of single element	Returns current values of editable elements. Note that it returns an **object** with name-value pairs, not a value itself. It allows to get data from several elements. If value of some editable is `null` or `undefined` it is excluded from result object. When param `isSingle` is set to **true** - it is supposed you have single element and will return value of editable instead of object. `$('#username, #fullname').editable('getValue');` `//result:` `{` `username: "superuser",` `fullname: "John"` `}` `//isSingle = true` `$('#username').editable('getValue', true);` `//result "superuser"`
hide()	*none*	Hides container with form
option(key, value)	• **key** *String|object* option name or object with several options • **value** *Mixed* option new value	Sets new option `$('.editable').editable('option', 'pk', 2);`
setValue(value, convertStr)	• **value** *Mixed* new value • **convertStr** *Boolean* whether to convert value from string to internal format	Sets new value of editable
show()	**closeAll** *Boolean* Whether to close all other editable containers when showing this one. Default true.	Shows container with form
submit(options)	**options** *Object* ◦ **url** *Object* url to submit data ◦ **data** *Object* additional data to submit ◦ **ajaxOptions** *Object* additional ajax options ◦ **error(obj)** *Function* error handler ◦ **success(obj,config)** *Function* success handler	This method collects values from several editable elements and submit them all to server. Internally it runs client-side validation for all fields and submits only in case of success. See creating new records for details. Since 1.5.1 `submit` can be applied to single element to send data programmatically. In that case `url`, `success` and `error` is taken from initial options and you can just call `$('#username').editable('submit')`.
toggle()	**closeAll** *Boolean* Whether to close all other editable containers when showing this one. Default true.	Toggles container visibility (show / hide)
toggleDisabled()	*none*	Toggles enabled / disabled state of editable element
validate()	*none*	Runs client-side validation for all matched editables `$('#username, #fullname').editable('validate');` `// possible result:` `{` `  username: "username is required",` `  fullname: "fullname should be minimum 3 letters length"` `}`

Event	Callback parameters	Description
hidden	• **event** *Object* event object • **reason** *String* Reason caused hiding. Can be `save|cancel|onblur|nochange|manual`	Fired when container was hidden. It occurs on both save or cancel. **Note:** Bootstrap popover has own `hidden` event that now cannot be separated from x-editable's one. The workaround is to check `arguments.length` that is always `2` for x-editable. `$('#username').on('hidden', function(e, reason) {` `  if(reason === 'save' \|\| reason === 'cancel') {` `  //auto-open next editable` `$(this).closest('tr').next().find('.editable').editable('show');` `  }` `});`
init since 1.2.0	• **event** *Object* event object • **editable** *Object* editable instance (as here it cannot accessed via data('editable'))	Fired when element was initialized by `$().editable()` method. Please note that you should setup `init` handler **before** applying `editable`. `$('#username').on('init', function(e, editable) {` `  alert('initialized ' + editable.options.name);` `});` `$('#username').editable();`
save	• **event** *Object* event object • **params** *Object* additional params ◦ **newValue** *Mixed* submitted value ◦ **response** *Object* ajax response	Fired when new value was submitted. You can use `$(this).data('editable')` to access to editable instance `$('#username').on('save', function(e, params) {` `  alert('Saved value: ' + params.newValue);` `});`
shown	**event** *Object* event object	Fired when container is shown and form is rendered (for select will wait for loading dropdown options). **Note:** Bootstrap popover has own `shown` event that now cannot be separated from x-editable's one. The workaround is to check `arguments.length` that is always `2` for x-editable. `$('#username').on('shown', function(e, editable) {` `  editable.input.$input.val('overwriting value of input..');` `});`

Note: you can modify `$.fn.editable.defaults` to set default options for all editable elements on the page.
For example, to force all elements submit via **PUT** method: `$.fn.editable.defaults.ajaxOptions = {type: "put"}`

Inputs

There are several *input types* supported by library. Each type may have additional configuration options. Input options are defined as well as other parameters of `$().editable()` method.
Currently supported:

- text
- textarea
- select
- date
- datetime
- dateui
- combodate
- html5types
- checklist

- wysihtml5
- typeahead
- typeaheadjs
- select2

text

Text input

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean	true	Whether to show `clear` button
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of input. Shown when input is empty.
tpl	string	<input type="text">	HTML template of input. Normally you should not change it.

```
<a href="#" id="username" data-type="text" data-pk="1">awesome</a>
<script>
$(function(){
    $('#username').editable({
        url: '/post',
        title: 'Enter username'
    });
});
</script>
```

textarea

Textarea input

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	input-large	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of input. Shown when input is empty.

Name	Type	Default	Description
rows	integer	7	Number of rows in textarea
tpl	string	<textarea> </textarea>	HTML template of input. Normally you should not change it.

```
<a href="#" id="comments" data-type="textarea" data-pk="1">awesome comment!</a>
<script>
$(function(){
    $('#comments').editable({
        url: '/post',
        title: 'Enter comments',
        rows: 10
    });
});
</script>
```

select

Select (dropdown)

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
prepend	string | array | object | function	false	Data automatically prepended to the beginning of dropdown list.
source	string | array | object | function	null	Source data for list. If **array** - it should be in format: `[{value: 1, text: "text1"}, {value: 2, text: "text2"}, ...]` For compability, object format is also supported: `{"1": "text1", "2": "text2" ...}` but it does not guarantee elements order. If **string** - considered ajax url to load items. In that case results will be cached for fields with the same source and name. See also `sourceCache` option. If **function**, it should return data in format above (since 1.4.0). Since 1.4.1 key `children` supported to render OPTGROUP (for **select** input only). `[{text: "group1", children: [{value: 1, text: "text1"}, {value: 2, text: "text2"}]}, ...]`
sourceCache since 1.2.0	boolean	true	if `true` and source is **string url** - results will be cached for fields with the same source. Usefull for editable column in grid to prevent extra requests.
sourceError	string	Error when loading list	Error message when list cannot be loaded (e.g. ajax error)

Name	Type	Default	Description
sourceOptions since 1.5.0	object|function	null	Additional ajax options to be used in $.ajax() when loading list from server. Useful to send extra parameters (`data` key) or change request method (`type` key).
tpl	string	<select> </select>	HTML template of input. Normally you should not change it.

```
<a href="#" id="status" data-type="select" data-pk="1" data-url="/post" data-title="Select
status"></a>
<script>
$(function(){
    $('#status').editable({
        value: 2,
        source: [
              {value: 1, text: 'Active'},
              {value: 2, text: 'Blocked'},
              {value: 3, text: 'Deleted'}
           ]
    });
});
</script>
```

date

Bootstrap-datepicker.
Description and examples: https://github.com/eternicode/bootstrap-datepicker.
For **i18n** you should include js file from here: https://github.com/eternicode/bootstrap-datepicker/tree/master/js/locales and set `language` option.
Since 1.4.0 date has different appearance in **popup** and **inline** modes.

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean|string	'x clear'	Text shown as clear date button. If `false` clear button will not be rendered.
datepicker	object	{ weekStart: 0, startView: 0, minViewMode: 0, autoclose: false }	Configuration of datepicker. Full list of options: http://bootstrap-datepicker.readthedocs.org/en/latest/options.html
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
format	string	yyyy-mm-dd	Format used for sending value to server. Also applied when converting date from `data-value` attribute. Possible tokens are: `d, dd, m, mm, yy, yyyy`
inputclass	string	null	CSS class automatically applied to input
tpl	string		HTML template of input. Normally you should not change it.

Name	Type	Default	Description
viewformat	string	null	Format used for displaying date. Also applied when converting date from element's text on init. If not specified equals to `format`

```
<a href="#" id="dob" data-type="date" data-pk="1" data-url="/post" data-title="Select
date">15/05/1984</a>
<script>
$(function(){
    $('#dob').editable({
        format: 'yyyy-mm-dd',
        viewformat: 'dd/mm/yyyy',
        datepicker: {
                weekStart: 1
           }
        });
});
</script>
```

datetime

Bootstrap-datetimepicker.
Based on smalot bootstrap-datetimepicker plugin. Before usage you should manually include dependent js and css:

```
<link href="css/datetimepicker.css" rel="stylesheet" type="text/css"></link>
<script src="js/bootstrap-datetimepicker.js"></script>
```

For **i18n** you should include js file from here: https://github.com/smalot/bootstrap-datetimepicker/tree/master/js/locales and set `language` option.

since 1.4.4

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean|string	'x clear'	Text shown as clear date button. If `false` clear button will not be rendered.
datetimepicker	object	{ }	Configuration of datetimepicker. Full list of options: https://github.com/smalot/bootstrap-datetimepicker
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
format	string	yyyy-mm-dd hh:ii	Format used for sending value to server. Also applied when converting date from `data-value` attribute. Possible tokens are: `d, dd, m, mm, yy, yyyy, h, i`
inputclass	string	null	CSS class automatically applied to input

Name	Type	Default	Description
tpl	string		HTML template of input. Normally you should not change it.
viewformat	string	null	Format used for displaying date. Also applied when converting date from element's text on init. If not specified equals to `format`

```
<a href="#" id="last_seen" data-type="datetime" data-pk="1" data-url="/post" title="Select date & time">15/03/2013 12:45</a>
<script>
$(function(){
    $('#last_seen').editable({
        format: 'yyyy-mm-dd hh:ii',
        viewformat: 'dd/mm/yyyy hh:ii',
        datetimepicker: {
                weekStart: 1
           }
        }
    });
});
</script>
```

dateui

jQuery UI Datepicker.
Description and examples: http://jqueryui.com/datepicker.
This input is also accessible as **date** type. Do not use it together with **bootstrap-datepicker** as both apply `$().datepicker()` method.
For **i18n** you should include js file from here: https://github.com/jquery/jquery-ui/tree/master/ui/i18n.

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean|string	'x clear'	Text shown as clear date button. If `false` clear button will not be rendered.
datepicker	object	{ firstDay: 0, changeYear: true, changeMonth: true }	Configuration of datepicker. Full list of options: http://api.jqueryui.com/datepicker
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
format	string	yyyy-mm-dd	Format used for sending value to server. Also applied when converting date from `data-value` attribute. Full list of tokens: http://docs.jquery.com/UI/Datepicker/formatDate
inputclass	string	null	CSS class automatically applied to input
tpl	string		HTML template of input. Normally you should not change it.

Name	Type	Default	Description
viewformat	string	null	Format used for displaying date. Also applied when converting date from element's text on init. If not specified equals to `format`

```
<a href="#" id="dob" data-type="date" data-pk="1" data-url="/post" data-title="Select
date">15/05/1984</a>
<script>
$(function(){
    $('#dob').editable({
        format: 'yyyy-mm-dd',
        viewformat: 'dd/mm/yyyy',
        datepicker: {
                firstDay: 1
           }
        }
    });
});
</script>
```

combodate

Combodate input - dropdown date and time picker.
Based on combodate plugin (included). To use it you should manually includemomentjs.

```
<script src="js/moment.min.js"></script>
```

Allows to input:

- only date
- only time
- both date and time

Please note, that format is taken from momentjs and **not compatible** with bootstrap-datepicker / jquery UI datepicker.
Internally value stored as `momentjs` object.

since 1.4.0

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
combodate	object	null	Configuration of combodate. Full list of options: http://vitalets.github.com/combodate/#docs
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
format	string	YYYY-MM-DD	Format used for sending value to server. Also applied when converting date from `data-value` attribute. See list of tokens in momentjs docs
inputclass	string	null	CSS class automatically applied to input

Name	Type	Default	Description
template	string	D / MMM / YYYY	Template used for displaying dropdowns.
tpl	string	<input type="text">	HTML template of input. Normally you should not change it.
viewformat	string	null	Format used for displaying date. Also applied when converting date from element's text on init. If not specified equals to `format`.

```
<a href="#" id="dob" data-type="combodate" data-pk="1" data-url="/post" data-value="1984-05-15"
data-title="Select date"></a>
<script>
$(function(){
    $('#dob').editable({
        format: 'YYYY-MM-DD',
        viewformat: 'DD.MM.YYYY',
        template: 'D / MMMM / YYYY',
        combodate: {
                minYear: 2000,
                maxYear: 2015,
                minuteStep: 1
           }
        }
    });
});
</script>
```

html5types

HTML5 input types. Following types are supported:

- password
- email
- url
- tel
- number
- range
- time

Learn more about html5 inputs:
http://www.w3.org/wiki/HTML5_form_additions
To check browser compatibility please see:
https://developer.mozilla.org/en-US/docs/HTML/Element/Input

since 1.3.0

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean	true	Whether to show `clear` button
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.

Name	Type	Default	Description
inputclass	string	null	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of input. Shown when input is empty.
tpl	string	depends on type	HTML template of input. Normally you should not change it.

```
<a href="#" id="email" data-type="email" data-pk="1">admin@example.com</a>
<script>
$(function(){
    $('#email').editable({
        url: '/post',
        title: 'Enter email'
    });
});
</script>
```

checklist

List of checkboxes. Internally value stored as javascript array of values.

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
prepend	string | array | object | function	false	Data automatically prepended to the beginning of dropdown list.
separator	string	','	Separator of values when reading from `data-value` attribute
source	string | array | object | function	null	Source data for list. If **array** - it should be in format: `[{value: 1, text: "text1"}, {value: 2, text: "text2"}, ...]` For compability, object format is also supported: `{"1": "text1", "2": "text2" ...}` but it does not guarantee elements order. If **string** - considered ajax url to load items. In that case results will be cached for fields with the same source and name. See also `sourceCache` option. If **function**, it should return data in format above (since 1.4.0). Since 1.4.1 key `children` supported to render OPTGROUP (for **select** input only). `[{text: "group1", children: [{value: 1, text: "text1"}, {value: 2, text: "text2"}]}, ...]`

Name	Type	Default	Description
sourceCache since 1.2.0	boolean	true	if `true` and source is **string url** - results will be cached for fields with the same source. Usefull for editable column in grid to prevent extra requests.
sourceError	string	Error when loading list	Error message when list cannot be loaded (e.g. ajax error)
sourceOptions since 1.5.0	object|function	null	Additional ajax options to be used in $.ajax() when loading list from server. Useful to send extra parameters (`data` key) or change request method (`type` key).
tpl	string		HTML template of input. Normally you should not change it.

```
<a href="#" id="options" data-type="checklist" data-pk="1" data-url="/post" data-title="Select
options"></a>
<script>
$(function(){
    $('#options').editable({
        value: [2, 3],
        source: [
              {value: 1, text: 'option1'},
              {value: 2, text: 'option2'},
              {value: 3, text: 'option3'}
           ]
    });
});
</script>
```

wysihtml5

Bootstrap wysihtml5 editor. Based on bootstrap-wysihtml5.
You should include **manually** distributives of `wysihtml5` and `bootstrap-wysihtml5` :

If you are Bootstrap 2

```
<link href="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.2/bootstrap-wysihtml5-0.0.2.css"
rel="stylesheet" type="text/css"></link>
<script src="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.2/wysihtml5-0.3.0.min.js"></script>
<script src="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.2/bootstrap-wysihtml5-0.0.2.min.js">
</script>
```

And also include `wysihtml5-0.0.2.js` from `inputs-ext` directory of x-editable:

```
<script src="js/inputs-ext/wysihtml5/wysihtml5-0.0.2.js"></script>
```

or you are Bootstrap 3

```
<link href="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.3/bootstrap-wysihtml5-0.0.3.css"
rel="stylesheet" type="text/css"></link>
<script src="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.3/wysihtml5-0.3.0.min.js"></script>
<script src="js/inputs-ext/wysihtml5/bootstrap-wysihtml5-0.0.3/bootstrap-wysihtml5-0.0.3.min.js">
</script>
```

And also include `wysihtml5-0.0.3.js` from `inputs-ext` directory of x-editable:

```
<script src="js/inputs-ext/wysihtml5/wysihtml5-0.0.3.js"></script>
```

Note: It's better to use fresh bootstrap-wysihtml5 from it's master branch as there is update for correct image insertion.

since 1.4.0

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	editable-wysihtml5	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of input. Shown when input is empty.
tpl	string	<textarea> </textarea>	HTML template of input. Normally you should not change it.
wysihtml5	object	{stylesheets: false}	Wysihtml5 default options. See https://github.com/jhollingworth/bootstrap-wysihtml5#options

```
<div id="comments" data-type="wysihtml5" data-pk="1"><h2>awesome</h2> comment!</div>
<script>
$(function(){
    $('#comments').editable({
        url: '/post',
        title: 'Enter comments'
    });
});
</script>
```

typeahead

Typeahead input (bootstrap 2 only). Based on Twitter Bootstrap 2typeahead.
Depending on `source` format typeahead operates in two modes:

- **strings**:
  When `source` defined as array of strings, e.g. `['text1', 'text2', 'text3' ...]`.
  User can submit one of these strings or any text entered in input (even if it is not matching source).
- **objects**:
  When `source` defined as array of objects, e.g. `[{value: 1, text: "text1"}, {value: 2, text: "text2"}, ...]`.
  User can submit only values that are in source (otherwise `null` is submitted). This is more like *dropdown* behavior.

since 1.4.1

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean	true	Whether to show `clear` button

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
prepend	string | array | object | function	false	Data automatically prepended to the beginning of dropdown list.
source	string | array | object | function	null	Source data for list. If **array** - it should be in format: `[{value: 1, text: "text1"}, {value: 2, text: "text2"}, ...]` For compability, object format is also supported: `{"1": "text1", "2": "text2" ...}` but it does not guarantee elements order. If **string** - considered ajax url to load items. In that case results will be cached for fields with the same source and name. See also `sourceCache` option. If **function**, it should return data in format above (since 1.4.0). Since 1.4.1 key `children` supported to render OPTGROUP (for **select** input only). `[{text: "group1", children: [{value: 1, text: "text1"}, {value: 2, text: "text2"}]}, ...]`
sourceCache since 1.2.0	boolean	true	if `true` and source is **string url** - results will be cached for fields with the same source. Usefull for editable column in grid to prevent extra requests.
sourceError	string	Error when loading list	Error message when list cannot be loaded (e.g. ajax error)
sourceOptions since 1.5.0	object|function	null	Additional ajax options to be used in $.ajax() when loading list from server. Useful to send extra parameters (`data` key) or change request method (`type` key).
tpl	string	<input type="text">	HTML template of input. Normally you should not change it.
typeahead	object	null	Configuration of typeahead. Full list of options.

```
<a href="#" id="country" data-type="typeahead" data-pk="1" data-url="/post" data-title="Input country"></a>
<script>
$(function(){
    $('#country').editable({
        value: 'ru',
        source: [
              {value: 'gb', text: 'Great Britain'},
              {value: 'us', text: 'United States'},
              {value: 'ru', text: 'Russia'}
           ]
    });
});
</script>
```

typeaheadjs

Typeahead.js input, based on Twitter Typeahead.
It is mainly replacement of typeahead in Bootstrap 3.

since 1.5.0

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
clear	boolean	true	Whether to show `clear` button
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of input. Shown when input is empty.
tpl	string	<input type="text">	HTML template of input. Normally you should not change it.
typeahead	object	null	Configuration of typeahead itself. Full list of options.

```
<a href="#" id="country" data-type="typeaheadjs" data-pk="1" data-url="/post" data-title="Input
country"></a>
<script>
$(function(){
    $('#country').editable({
        value: 'ru',
        typeahead: {
            name: 'country',
            local: [
                {value: 'ru', tokens: ['Russia']},
                {value: 'gb', tokens: ['Great Britain']},
                {value: 'us', tokens: ['United States']}
            ],
            template: function(item) {
                return item.tokens[0] + ' (' + item.value + ')';
            }
        }
    });
});
</script>
```

select2

Select2 input. Based on amazing work of Igor Vaynberg https://github.com/ivaynberg/select2.
Please see original select2 docs for detailed description and options.

You should manually download and include select2 distributive:

```
<link href="select2/select2.css" rel="stylesheet" type="text/css"></link>
<script src="select2/select2.js"></script>
```

To make it **bootstrap-styled** you can use css from here:

```
<link href="select2-bootstrap.css" rel="stylesheet" type="text/css"></link>
```

Note: currently `autotext` feature does not work for select2 with `ajax` remote source.
You need initially put both `data-value` and element's text youself:

```
<a href="#" data-type="select2" data-value="1">Text1</a>
```

since 1.4.1

- Options
- Example

Options can be defined via javascript `$().editable({...})` or via `data-*` html attributes.

Name	Type	Default	Description
escape since 1.5.0	boolean	true	If `true` - html will be escaped in content of element via $.text() method. If `false` - html will not be escaped, $.html() used. When you use own `display` function, this option obviosly has no effect.
inputclass	string	null	CSS class automatically applied to input
placeholder	string	null	Placeholder attribute of select
select2	object	null	Configuration of select2. Full list of options.
source	array|string|function	null	Source data for select. It will be assigned to select2 `data` property and kept here just for convenience. Please note, that format is different from simple `select` input: use 'id' instead of 'value'. E.g. `[{id: 1, text: "text1"}, {id: 2, text: "text2"}, ...]`.
tpl	string	<input type="hidden">	HTML template of input. Normally you should not change it.
viewseparator	string	', '	Separator used to display tags.

```
<a href="#" id="country" data-type="select2" data-pk="1" data-value="ru" data-url="/post" data-
title="Select country"></a>
<script>
$(function(){
    //local source
    $('#country').editable({
        source: [
              {id: 'gb', text: 'Great Britain'},
              {id: 'us', text: 'United States'},
              {id: 'ru', text: 'Russia'}
           ],
        select2: {
           multiple: true
        }
    });
    //remote source (simple)
    $('#country').editable({
        source: '/getCountries',
        select2: {
            placeholder: 'Select Country',
            minimumInputLength: 1
        }
    });
    //remote source (advanced)
    $('#country').editable({
        select2: {
            placeholder: 'Select Country',
            allowClear: true,
            minimumInputLength: 3,
            id: function (item) {
                return item.CountryId;
            },
            ajax: {
                url: '/getCountries',
                dataType: 'json',
                data: function (term, page) {
                    return { query: term };
                },
                results: function (data, page) {
                    return { results: data };
                }
            },
            formatResult: function (item) {
                return item.CountryName;
            },
            formatSelection: function (item) {
                return item.CountryName;
            },
            initSelection: function (element, callback) {
                return $.get('/getCountryById', { query: element.val() }, function (data) {
                    callback(data);
                });
            }
        }
    });
});
</script>
```

Global templates

Some templates are defined globally and affect all editable inputs. You can overwrite it for your needs.

$.fn.editableform.template

Form template. Must contain `FORM` tag and classes `.control-group`, `.editable-input`, `.editable-buttons`, `.editable-error-block`.
Default value:

```
<form class="form-inline editableform">
    <div class="control-group">
         <div><div class="editable-input"></div><div class="editable-buttons"></div></div>
         <div class="editable-error-block"></div>
    </div>
</form>
```

$.fn.editableform.buttons

Buttons template. Automatically inserted into `.editable-buttons`. Must contain classes `.editable-submit` and `.editable-cancel`.
Default value:

```
<button type="submit" class="editable-submit">ok</button>
<button type="button" class="editable-cancel">cancel</button>
```

$.fn.editableform.loading

Loading DIV template. Default value:

```
<div class="editableform-loading"></div>
```

Creating new record

Creating **new record** is a bit problem for editable interface because primary key (record id) is yet*unknown* and in-place-editing has no related object on backend. In that case element will store new value inside and will not submit anything to server until you do it manually. Such behavior applied when `send` option is `'never'` or `send='auto'` and `pk` is empty. To submit manually several editable fields use `submit()` method. Try example below to see how it works.

Live example - creating new record:

Username	
First name	
Group	
Date of birth	
Comments	

How it works:

Initialization performed without `pk` option:

```
//init editables
$('.myeditable').editable({
    url: '/post' //this url will not be used for creating new user, it is only for update
});

//make username required
$('#new_username').editable('option', 'validate', function(v) {
    if(!v) return 'Required field!';
});

//automatically show next editable
$('.myeditable').on('save.newuser', function(){
    var that = this;
    setTimeout(function() {
        $(that).closest('tr').next().find('.myeditable').editable('show');
    }, 200);
});
```

When you enter data first time - nothing sent to server, but all values are stored.
When you click on save button - `submit()` method is called. It runs validation for all fields and if no errors - submits to server.
Two callbacks `success` and `error` are passed into `submit()`. Success applied if **http response = 200 OK**, otherwise called error.
Inside `success` you can update editables with new `pk` received from server and they will start working in regular way (submit individually).

Save button click:

```
$('#save-btn').click(function() {
   $('.myeditable').editable('submit', {
       url: '/newuser',
       ajaxOptions: {
           dataType: 'json' //assuming json response
       },
       success: function(data, config) {
           if(data && data.id) {  //record created, response like {"id": 2}
               //set pk
               $(this).editable('option', 'pk', data.id);
               //remove unsaved class
               $(this).removeClass('editable-unsaved');
               //show messages
               var msg = 'New user created! Now editables submit individually.';
               $('#msg').addClass('alert-success').removeClass('alert-error').html(msg).show();
               $('#save-btn').hide();
               $(this).off('save.newuser');
           } else if(data && data.errors){
               //server-side validation error, response like {"errors": {"username": "username
already exist"} }
               config.error.call(this, data.errors);
           }
       },
       error: function(errors) {
           var msg = '';
           if(errors && errors.responseText) { //ajax error, errors = xhr object
               msg = errors.responseText;
           } else { //validation error (client-side or server-side)
               $.each(errors, function(k, v) { msg += k+": "+v+"<br>"; });
           }
           $('#msg').removeClass('alert-success').addClass('alert-error').html(msg).show();
       }
   });
});
```

There are three possible error scenarios:

- **client-side validation error**
  `error` callback applied with parameter containing field names and errors. E.g.

  ```
  error: function(data) {
  // data is: { username: 'Required field!' }
  ...
  }
  ```

- **server-side validation error**
  In that case response status is **200 OK** but it's body contains error information.
  Initially `success` is called, but you can check response inside and if needed apply `error` (see example above).
  To do it simply there is second parameter of success that is exactly config object.

  ```
  success: function(data, config) {
     if(data.id) {  //record created, response like {"id": 2}
         //proceed success...
     } else if(data.errors){
         //server-side validation error, response like {"errors": {"username": "username
  already exist"} }
         //call error
         config.error.call(this, data.errors);
     }
  }
  ```

- **ajax error, http status != 200**
  `error` callback called with xhr object as parameter.

Reset button click:

```
$('#reset-btn').click(function() {
    $('.myeditable').editable('setValue', null)  //clear values
        .editable('option', 'pk', null)          //clear pk
        .removeClass('editable-unsaved');        //remove bold css

    $('#save-btn').show();
    $('#msg').hide();
});
```

Note: you can use this approach not only for creating new record but also for submitting several fields of existing record at once.

Backend samples

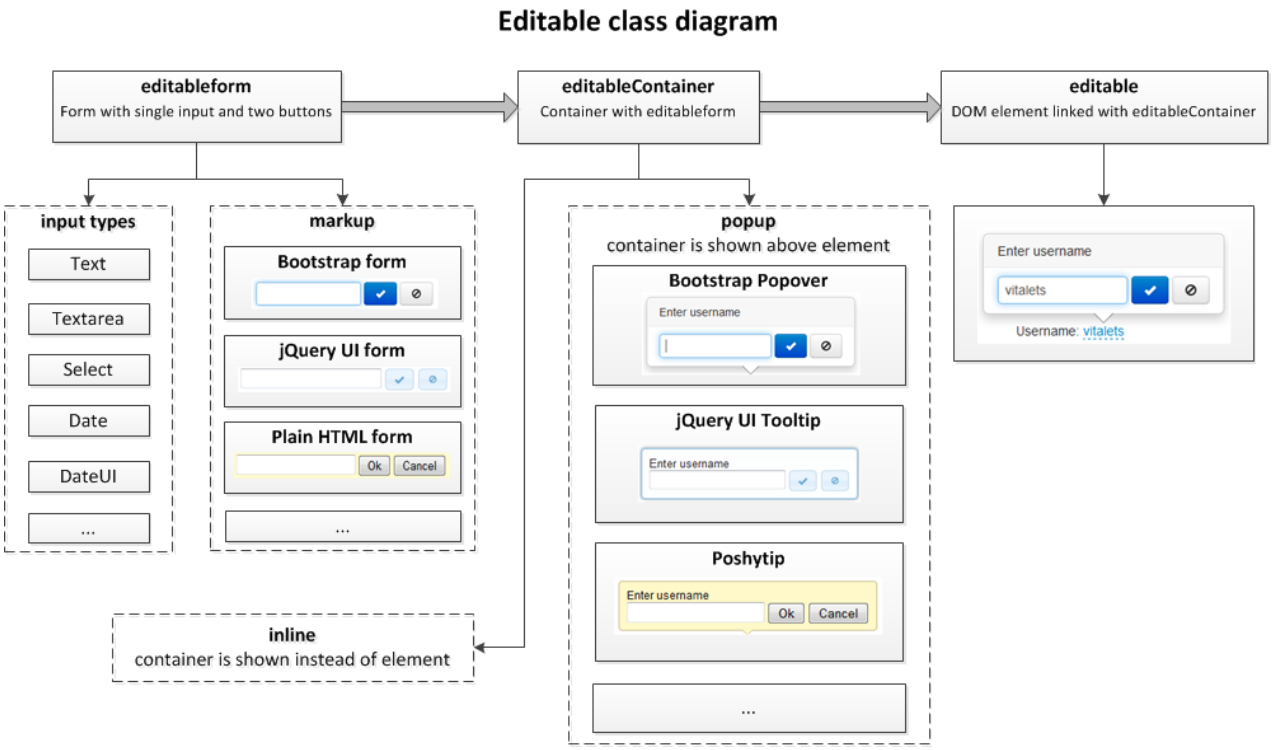
X-editable has no limitation to backend side, you can write it on any language you prefer.
For easy starting have a look on sample projects below:

If you'd like to share you sample please feel free to send pull request on `gh-pages-dev` branch.

Class diagram

Here a short description how it works internally helping you to extend library for your needs.
There are a few classes working together for editable. Base class **editableform** is form with single input and two buttons. Input can be one of several **input types** (text, textarea, etc). Form **markup** is performed in plain HTML or can be extended by some library (bootstrap, jquery-ui). **editableContainer** includes editableform and shows it in **popup** or **inline** mode. When popup it can use any container for display

(popover, tooltip, etc). Final class **editable** binds editableContainer to DOM element storing value and being updated after user submits new value.



If you have questions, ideas or find a bug - please feel free to open issue onX-editable github page.