In-place editing is one of the hallmarks of Web 2.0.style applications.

In-place editing is about taking non-editable content, such as a <p>, <h1>, or , and letting the user edit its contents by simply clicking it.

This turns the static element into an editable zone *eithersinglelineormultiline* and pops up submit and cancel buttons *orlinks, dependingonyouroptions* for the user to commit or roll back the modification.

It then synchronizes the edit on the server side through Ajax and makes the element non-editable again.

To use script.aculo.us's in-place editing capabilities, you'll need to load the controls.js and effects.js modules along with the prototype.js module. So, your minimum loading for script.aculo.us will look like this −

```
<script type="text/javascript" src="/javascript/prototype.js"></script>
<script type="text/javascript" src="/javascript/scriptaculous.js?
load=effects,controls"></script>
```

Creating an In-Place Text Editor

The whole construction syntax is as follows −

```
new Ajax.InPlaceEditor(element, url [ , options ] )
```

The constructor for the Ajax.InPlaceEditor accepts three parameters −

- The target element can either be a reference to the element itself or the id of the target element.
- The second parameter to the Ajax.InPlaceEditor specifies the URL of a server-side script that is contacted when an edited value is completed.
- The usual options hash.

Options

You can use one or more of the following options while creating your Ajax.InPlaceEditor object.

Option	Description
okButton	A Boolean value indicating whether an "ok" button is to be shown or not. Defaults to true.
okText	The text to be placed on the ok button. Defaults to "ok".
cancelLink	A Boolean value indicating whether a cancel link should be displayed. Defaults to true.
cancelText	The text of the cancel link. Defaults to "cancel".
savingText	A text string displayed as the value of the control while the save operation *therequestinitiatedbyclickingtheokbutton* is processing. Defaults to "Saving".
clickToEditText	The text string that appears as the control "tooltip" upon mouse-over.
rows	The number of rows to appear when the edit control is active. Any number greater than 1 causes a text area element to be used rather than a text field element. Defaults to 1.

cols	The number of columns when in active mode. If omitted, no column limit is imposed.
size	Same as cols but only applies when rows is 1.
highlightcolor	The color to apply to the background of the text element upon mouse-over. Defaults to a pale yellow.
highlightendcolor	The color to which the highlight color fades to as an effect. **Note** − support seems to be spotty in some browsers.
loadingText	The text to appear within the control during a load operation. The default is "Loading".
loadTextURL	Specifies the URL of a server-side resource to be contacted in order to load the initial value of the editor when it enters active mode. By default, no backend load operation takes place and the initial value is the text of the target element.
externalControl	An element that is to serve as an "external control" that triggers placing the editor into an active mode. This is useful if you want another button or other element to trigger editing the control.
ajaxOptions	A hash object that will be passed to the underlying Prototype Ajax object to use as its options hash.

Callback Options

Additionally, you can use any of the following callback functions in the options parameter

Function	Description
onComplete	A JavaScript function that is called upon successful completion of the save request. The default applies a highlight effect to the editor.
onFailure	A JavaScript function that is called upon failure of the save request. The default issues an alert showing the failure message.
callback	A JavaScript function that is called just prior to submitting the save request in order to obtain the query string to be sent to the request. The default function returns a query string equating the query parameter "value" to the value in the text control.

CSS Styling and DOM id Options

You can also use one the following options to control the behavior of in place editor −

Option	Description
savingClassName	The CSS class name applied to the element while the save operation is in progress. This class is applied when the request to the saving URL is made, and is removed when the response is returned. The default value is "inplaceeditor-saving".
formClassName	The CSS class name applied to the form created to contain the editor element. Defaults to "inplaceeditor-form".
formId	The id applied to the form created to contain the editor element.

Example

```
<html>
   <head>
      <title>Simple Ajax Auto-completer Example</title>

      <script type="text/javascript" src="/javascript/prototype.js"></script>
      <script type="text/javascript"  src="/javascript/scriptaculous.js?
load=effects,controls"></script>
      <script type="text/javascript">

         window.onload = function() {
            new Ajax.InPlaceEditor('theElement', '/script.aculo.us/transform.php',
{formId: 'whatever', okText: 'Upper me!', cancelText: 'Never mind'});
         }
      </script>
   </head>
   <body>

      <p>Click over the "Click me!" text and then change text and click OK.</p>
      <p>Try this example with different options.</p>

      <div >
         Click me!
      </div>

   </body>
</html>
```

When displayed, click and edit the text. This rather trivial PHP script converts the value of a query parameter with the key "value" to its uppercase equivalent, and writes the result back to the response.

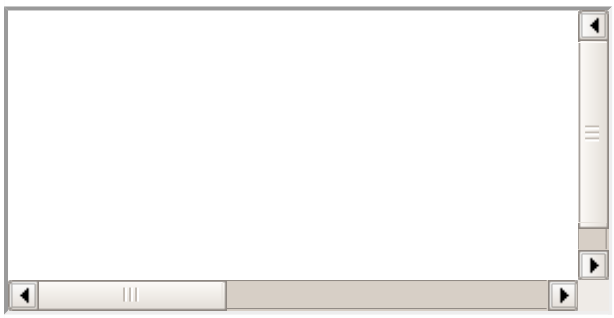
Here is the content of transform.php script.

```
<?php

   if( isset($_REQUEST["value"]) )
   {
      $str = $_REQUEST["value"];
      $str = strtoupper($str);
      echo "$str";
   }

?>
```

This will produce following result −



The In-Place Collection Editor Options

There is one more object called *Ajax.InPlaceCollectionEditor,* which supports in-place editing and gives you the option to select a value from the given options.

The whole construction syntax is as follows −

```
new Ajax.InPlaceCollectionEditor(element, url [ , options ] )
```

The constructor for the Ajax.InPlaceCollectionEditor accepts three parameters:

- The target element can either be a reference to the element itself or the id of the target element
- The second parameter to the Ajax.InPlaceEditor specifies the URL of a server-side script that is contacted when an edited value is completed.
- The usual options hash.

Options

Aside from the addition of the collection option, the list of options for the In-Place Collection Editor is a subset of the options inherited from the In-Place Text Editor.

Option	Description
okButton	A Boolean value indicating whether an "ok" button is to be shown or not. Defaults to true.
okText	The text to be placed on the ok button. Defaults to "ok".
cancelLink	A Boolean value indicating whether a cancel link should be displayed. Defaults to true.
cancelText	The text of the cancel link. Defaults to "cancel".
savingText	A text string displayed as the value of the control while the save operation *therequestinitiatedbyclickingtheokbutton* is processing. Defaults to "Saving".
clickToEditText	The text string that appears as the control "tooltip" upon mouse-over.
Highlightcolor	The color to apply to the background of the text element upon mouse-over. Defaults to a pale yellow.
Highlightendcolor	The color to which the highlight color fades to as an effect. **Note** − support seems to be spotty in some browsers.
Collection	An array of items that are to be used to populate the select element options.
loadTextUrl	Specifies the URL of a server-side resource to be contacted in order to load the initial value of the editor when it enters active mode. By default, no backend load operation takes place and the initial value is the text of the target element. In order for this option to be meaningful, it must return one of the items provided in the collection option to be set as the initial value of the select element.
externalControl	An element that is to serve as an "external control" that triggers placing the editor into active mode. This is useful if you want another button or other element to trigger editing the control.
ajaxOptions	A hash object that will be passed to the underlying Prototype Ajax object to use as its options hash.

Callback Options

Additionally, you can use any of the following callback functions in the options parameter −

Function	Description
onComplete	A JavaScript function that is called upon successful completion of the save request. The default applies a highlight effect to the editor.
onFailure	A JavaScript function that is called upon failure of the save request. The default issues an alert showing the failure message.

CSS Styling and DOM id Options

You can also use one the following options to control the behavior of in-place editor −

Option	Description
savingClassName	The CSS class name applied to the element while the save operation is in progress. This class is applied when the request to the saving URL is made, and is removed when the response is returned. The default value is "inplaceeditor-saving".
formClassName	The CSS class name applied to the form created to contain the editor element. Defaults to "inplaceeditor-form".
formId	The id applied to the form created to contain the editor element.

Example

```
<html>
   <head>
      <title>Simple Ajax Auto-completer Example</title>

      <script type="text/javascript" src="/javascript/prototype.js"></script>
      <script type="text/javascript"  src="/javascript/scriptaculous.js?
load=effects,controls"></script>
      <script type="text/javascript">

         window.onload = function() {
            new Ajax.InPlaceCollectionEditor('theElement',
'/script.aculo.us/transform.php', {formId: 'whatever', okText: 'Upper me!', cancelText:
'Never mind', collection: ['one','two','three','four','five']});
         }
      </script>
   </head>
   <body>

      <p>Click over the "Click me!" text and then change text and click OK.</p>
      <p>Try this example with different options.</p>

      <div >
         Click me!
      </div>

   </body>
</html>
```

Here is the content of the transform.php script.

```
<?php

   if( isset($_REQUEST["value"]) )
   {
      $str = $_REQUEST["value"];
      $str = strtoupper($str);
      echo "$str";
```

```
    }

?>
```

When displayed, click and select one of the displayed options. This rather trivial PHP script converts the value of the query parameter with the key "value" to its uppercase equivalent, and writes the result back to the response.

Use our online compiler for a better understanding of the code with different options discussed in the above table.

This will produce following result −