

# {input}

Erstellt ein neues Eingabe-Element. Kann im Adminbereich auf eigenen/angepassten Reitern eingesetzt werden.

Attribut	Typ	Erforderlich	Beschreibung
type	string	Ja	Typ des Plugins (z.B. "button", "color" usw.)
name	string	Ja	Name des Feldes
title	string	Ja	Beschriftung
short	string	Nein	Zusätzliche Kurzbeschreibung
no_auto	boolean	Nein	Mit no_auto="1" wird das automatische Speichern/Laden in bzw. aus dem extra-Feld deaktiviert
standalone	boolean	Nein	Mit standalone="1" wird das Inputplugin wie ein einfaches Formularfeld eingefügt, ohne Formatierung und den title

## Beispiel

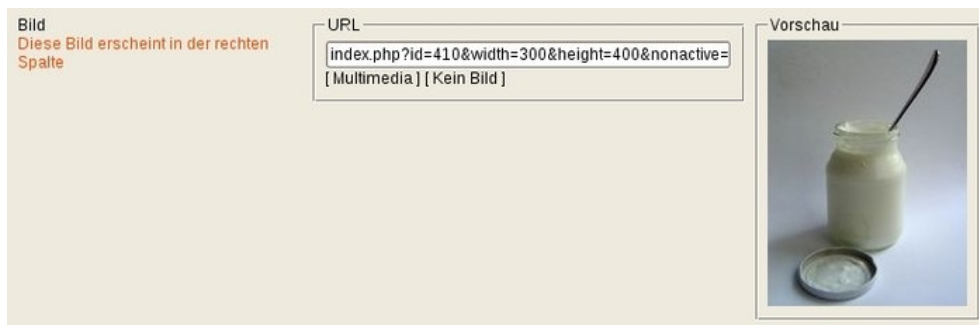
QuelltextSmarty Code:

1. {\* Eingabefeld für ein Bild erzeugen \*}
2. {input version=5 type="image" name="bild\_rechts" title="Bild" short="Diese Bild erscheint in der rechten Spalte"}

{\* Eingabefeld für ein Bild erzeugen \*}

{input version=5 type="image" name="bild\_rechts" title="Bild" short="Diese Bild erscheint in der rechten Spalte"}

**Ausgabe:**



## Die HTML-Datei

wir erstellen zuerst eine HTML-Datei

jede HTML Datei besitzt 2 Bereiche

### Bereich 1 Smarty+HTML

QuelltextSmarty Code:

1. <form name="extra">
2. <div align="center">
3. <table class="table">
4. <tr>
5. <td class="cell">

```

6.      {input version=5 type="button" name="input_test" title="Dies ist ein button"
onclick="javascript:test()}}
7.      </td>
8.      </tr>
9. </table>
10. </div>
11. </form>

```

```

<form name="extra">
<div align="center">
<table class="table">
  <tr>
    <td class="cell">
      {input version=5 type="button" name="input_test" title="Dies ist ein button" onclick="javascript:test()}}
    </td>
  </tr>
</table>
</div>
</form>

```

Achten Sie darauf, dass das Formular auf jeden Fall den Namen *extra* bekommt  
<div> und <table> sind für die einheitliche Gestaltung optional

## Bereich 2 Javascript

Ab Version 5 werden bei Seiten die extra-Felder automatisch gespeichert und geladen.

Es muss nur die unload\_extra() Funktion die do\_unload bzw. do\_unload\_extra Funktion aufgenommen werden.

QuelltextJavaScript Code:

```

1. // Für die Erweiterung des Informationsreiters
2.
3. <script language="javascript">
4. {literal}
5. function do_load_extra(){
6.
7. function do_unload_extra()
8. {
9.   window.parent.unload_extra()
10. }
11. {/literal}
12. </script>
13.
14.
15. //Für einen neuen Reiter
16.
17. <script language="javascript">
18. {literal}
19. function do_load()
20. {
21. }
22.
23. function do_unload()
24. {
25.   window.parent.unload_extra();

```

```
26. }
27. {/literal}
28. </script>
29.
```

// Für die Erweiterung des Informationsreiters

```
<script language="javascript">
{literal}
function do_load_extra(){

function do_unload_extra()
{
    window.parent.unload_extra()
}
{/literal}
</script>
```

//Für einen neuen Reiter

```
<script language="javascript">
{literal}
function do_load()
{

function do_unload()
{
    window.parent.unload_extra();
}
{/literal}
</script>
```

Wenn Sie Input-Plugins an anderen Stellen verwenden, z.B. auf einem Konfigurationsreiter in den Globalen Einstellungen können die Felder nicht automatisch abgespeichert werden.

Die Inhalte der Input-Felder können in dieser Version einheitlich mit den Funktionen `set_input_value()` und `get_input_value()` geladen und gespeichert werden.

QuelltextJavaScript Code:

```
1. function do_load()
2. {
3.     set_input_value('menue_color', window.parent.get_conf('administration', 'menue_color'));
4. }
5.
6. function do_unload()
7. {
8.     window.parent.set_conf('administration', 'menue_color', get_input_value('menue_color'));
9. }
```

```
function do_load()
{
    set_input_value('menue_color', window.parent.get_conf('administration', 'menue_color'));
}
```

```
function do_unload()
{
```

```
    window.parent.set_conf('administration', 'menue_color', get_input_value('menue_color'));  
}
```

## do\_load

In diesem Teil werden die Werte beim Speichern in die DB geschrieben

## do\_unload

Hier werden, wenn vorhanden, gespeicherte Werte an das Formular übergeben

Mit Dojo generierte Input-Felder können nicht vollständig über die Eigenschaften ihrer ursprünglichen HTML Objekte manipuliert werden. Hierfür ist es manchmal notwendig das dazugehörige Dojo Objekt zu ermitteln und dieses dann zu manipulieren. Die Funktion hierfür ist `get_input_object()`. Übergeben wird der Name des Input-Feldes.

### QuelltextJavaScript Code:

```
1. var obj = get_input_object('menue_color');
```

```
var obj = get_input_object('menue_color');
```

Soll ein Attribut geändert werden, verwendet man die `attr()` Funktion. Nur so aktualisiert Dojo seine Objekte.

### QuelltextJavaScript Code:

```
1. obj.attr('disabled', true); // ändert einen Wert
```

```
2.
```

```
3. obj.attr({disabled : true, label : 'Neuer Text'}); // ändert mehrere Werte auf einmal
```

```
obj.attr('disabled', true); // ändert einen Wert
```

```
obj.attr({disabled : true, label : 'Neuer Text'}); // ändert mehrere Werte auf einmal
```

Auslesen von Eigenschaften ist hingegen wie gewohnt machbar. Eigenschaften oder Arrays wie `checked`, `selected`, `options`, usw stehen nun auch Verfügung.

### QuelltextJavaScript Code:

```
1. obj.checked;
```

```
2. obj.selected;
```

```
3. obj.options;
```

```
obj.checked;
```

```
obj.selected;
```

```
obj.options;
```

Erstellt mit  
EGOTEC®  
Internet:  
[www.egotec.com](http://www.egotec.com)  
© EGOTEC  
GmbH

EGOTEC GmbH  
Alte Neckarelzer Straße 24  
D-74821 Mosbach

Tel: +49 (0)6261 /  
6743-0  
Fax: +49 (0)6261  
/ 6743-29  
E-Mail:  
[info@egotec.com](mailto:info@egotec.com)