

# Aufmacher

Stefan Freimark

## Eine bessere Welt ist möglich



Dateien auf CD

**Haben Sie sich auch schon mal über eine Website geärgert? Da will man Ihnen etwas verkaufen, aber der Anbieter legt Ihnen Steine in den Weg. Sei es ein unübersichtliches Formular, das selbst Ihre Schuhgröße wissen möchte oder unklare Fehlermeldungen: Vermeiden Sie die Fehler der anderen, indem Sie mit wenigen einfachen Regeln das Einkaufserlebnis für Ihre Besucher verbessern.**

Sie kennen das: Sie haben ein langes Formular ausgefüllt, und ein Moment der Unachtsamkeit genügt, um mit einem Klick alles zu löschen. Oder Sie zerbrechen sich den Kopf über eine angebotene Option, wie auf einer Bestellseite der Bahn (siehe Abbildung): Natürlich möchte ich, dass meine BahnCard noch heute in die Post geht. Wird mein Antrag nur schneller bearbeitet, wenn ich das ankreuze? Kostet das extra?

Dabei ist es so einfach, leicht verständliche Formulare zu programmieren. Der Kunde möchte Ihnen Geld geben, und warum sollten Sie ihn davon abhalten? Machen Sie es ihm doch besser so leicht wie möglich! Auch wenn es im Folgenden hauptsächlich um E-Commerce geht: Die Hinweise gelten ebenfalls für Newsletter-Bestellungen und andere Anwendungsbeispiele.

**Frustpotenzial senken.** Zunächst sollten Sie die Stolperfallen eliminieren, die den größten Schaden verursachen. Das Schlimmste, was dem Benutzer passieren kann, ist der Verlust aller eingegebenen Daten. Sie können zwei Dinge tun, um dem entgegen zu wirken: Schmeißen Sie erstens den Reset-Button raus und verwenden Sie zweitens ein Affenformular.

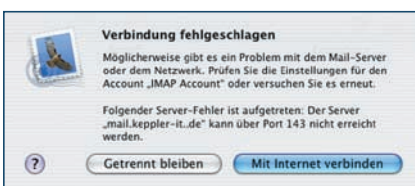


**Verwirrend:** Kostet der 24-Stunden-Service extra? Wenn nein, weswegen sollte der Kunde darauf verzichten? Lassen Sie überflüssige Fragen weg.

Der Reset-Button ist ein Selbsterstörungsknopf, da er mit einem Schlag alle Eingaben löscht. Und wie oft kommt es vor, dass ein Benutzer alles, was er eingetippt hat, löschen möchte? Lassen Sie den Reset-Button lieber weg, er muss nicht zu einem Formular gehören. Im Gegenteil, ein gutes Formular zeichnet sich dadurch aus, dass es *keinen* Reset-Button hat. Denken Sie daran: Machen Sie es dem Besucher so einfach wie möglich, das Formular abzuschicken. Unabhängig von Formularen gilt außerdem: Wenn eine Aktion Daten löscht, fragen Sie vorher beim Benutzer nach, ob er das wirklich möchte.

Wenn das Formular abgeschickt wurde und in der Formularprüfung hängen bleibt, sollte der Besucher natürlich nicht alles noch einmal eingeben müssen, weil ihm jetzt leere Felder präsentiert werden. Stellen Sie sich vor, Sie füllen solch ein Formular aus und machen, trotz maximaler Konzentration, wieder einen Fehler. Wie oft würden Sie wohl Anlauf nehmen und das Formular wieder und wieder ausfüllen? Zehnmal? Wahrscheinlich nicht. Hier hilft ein so genanntes Affenformular. Das Affenformular heißt so, weil eine Million Affen es immer wieder abschicken können, ohne dass die Eingaben verloren gehen.

Angenommen, Sie nehmen Ihre For-



**Klare Fehlermeldung:** Dieser Hinweis aus „Mail“ von Apple gibt Tipps zur Problemlösung.

mularprüfung mit PHP vor. Die Variablen aus dem Formular kommen dann in den Arrays `$_GET` oder `$_POST` an. Wenn die Seite vom Webserver neu generiert wird, können Sie die Eingabefelder mit den Werten aus diesen Arrays füllen. Schreiben Sie beispielsweise:

```
<input name="nachname" id="nachname" type="text" size="32" maxlength="255" value="<?php print $_POST['nachname']; ?>" />
```

Dies geht in ähnlicher Weise auch bei Radio-Buttons und Checkboxes (Listing 1).

**Wichtig:** Bei Radio-Buttons und Checkboxes müssen Sie identische *name*-Attribute vergeben, damit die Button-Gruppe funk-

### Listing 1

```
<label for="anrede_frau">Frau</label>
<input name="anrede" id="anrede_frau" type="radio" value="Frau" <?php if ($_POST['anrede'] == "Frau") { print "checked=\"checked\""; } ?> /><br />
<label for="anrede_herr">Herr</label>
<input name="anrede" id="anrede_herr" type="radio" value="Herr" <?php if ($_POST['anrede'] == "Herr") { print "checked=\"checked\""; } ?> />

<label for="zimmer_option1">Seeblick</label>
<input name="optionen[]" id="zimmer_option1" type="checkbox" value="Seeblick" <?php if (in_array("Seeblick", $_POST['optionen'])) { print "checked=\"checked\""; } ?> /><br />
<label for="zimmer_option2">Klimaanlage</label>
<input name="optionen[]" id="zimmer_option2" type="checkbox" value="Klimaanlage" <?php if (in_array("Klimaanlage", $_POST['optionen'])) { print "checked=\"checked\""; } ?> />
```

tioniert. Die *id*-Attribute müssen dagegen unterschiedlich sein, weil jede ID nur einmal im Dokument vorkommen darf (außerdem nehmen Sie mit dem *label*-Tag Bezug auf eine bestimmte ID). Für die Beschriftung vor allem von *input*-Feldern sollten Sie das *label*-Tag benutzen. Eine genauere Erläuterung würde zu weit das Thema Usability (Benutzerfreundlichkeit) verlassen; sie ist besser bei Accessibility (Zugänglichkeit) aufgehoben. Bei Auswahllisten schreiben Sie Code wie in Listing 2.

Für Textfelder mit `<textarea>` sieht Ihr Code so aus:

```
<textarea name="textfield" id="textfield" cols="45" rows="10"><?php print $_POST["textfield"]; ?>
</textarea>
```

Diese beiden Möglichkeiten, Verzicht auf den Reset-Button und ein Affenformular, bewahren den Benutzer vor Datenverlust. Für ein einfaches Formular können Sie jedoch noch mehr unternehmen.

**Vermeidung von Denksport.** Wenn Sie sich niemals überlegen „Das kann sich der Benutzer schon denken“, dann läuft etwas verkehrt. Einer der wichtigsten Grundsätze der Benutzerfreundlichkeit lautet: „Don’t make me think.“ Lass’ mich nicht drüber nachdenken (der Satz stammt von Steve Krug; siehe Literaturliste). Wenn Sie sich nur eine Usability-Regel merken möchten, dann sollte es diese sein. Auch wenn es nur Kleinigkeiten sind: Jede Kleinigkeit stört den Besucher bei seiner Aufgabe, erfolgreich ein Formular abzu-

### Listing 2

```
<?php $liste_von = $_POST["von_zeit"]; ?>
<select name="von_zeit" id="von_zeit" size="1">
<option value="0" <?php if ($liste_von == "0") { print "selected=\"selected\""; } ?>>
 (Bitte ausw&auml;hlen)</option>
<option value="09" <?php if ($liste_von == "09") { print "selected=\"selected\""; } ?>>
 09:00 Uhr</option>
<option value="10" <?php if ($liste_von == "10") { print "selected=\"selected\""; } ?>>
 10:00 Uhr</option>
<option value="11" <?php if ($liste_von == "11") { print "selected=\"selected\""; } ?>>
 11:00 Uhr</option>
</select>
```

schicken. Und das ist das Einzige, was Sie wollen – also tun Sie alles, um den Besucher zur Kasse zu tragen! Es sollte selbstverständlich sein, dass Pflichtfelder auch als solche zu erkennen sind, damit der Besucher weiß, welche Eingaben notwendig und welche optional sind. Wie Sie Pflichtfelder hervorheben, ist egal: Schreiben Sie die Beschriftung fett oder markieren Sie sie mit einem Sternchen (vergessen Sie aber nicht, diese Fußnote aufzuklären, indem Sie im Blickfeld des Besuchers schreiben, dass es sich hierbei um Pflichtfelder handelt). Beschränken Sie sich bei der Anzahl der Pflichtfelder auf das Nötigste: Wenn Sie zu viele Daten abfragen, wird sich der Besucher überlegen, ob es den Aufwand und die Preisgabe seiner Daten wert ist. Je mehr Sie wissen möchten, desto wahrscheinlicher wird es, dass sich der Kunde abwendet. Müssen Sie bei einer Newsletter-Anmeldung unbedingt das Geburtsdatum und das Haushaltsnettoeinkommen des Empfängers wissen?

Wenn der Besucher vergisst, eines der Pflichtfelder auszufüllen, weisen Sie ihn bestimmt mit einer Fehlermeldung auf diesen Umstand hin. Versäumen Sie auch hier nicht, den Ort zu markieren, an dem der Fehler aufgetreten ist: Lassen Sie den Besucher nicht suchen, sondern heben Sie zum Beispiel die Feldbezeichnung farbig hervor. In der Abbildung sehen Sie eine Möglichkeit hierfür.

Fehlermeldungen beschreiben den Fehler. Klingt trivial – allerdings habe ich schon zu viele Hinweise der Sorte „von Programmierern für Programmierer“ gesehen, um es hier unerwähnt zu lassen. Schreiben Sie nichts von „Exception 0815“ und belassen Sie es nicht bei einem lapidaren „name-1“. Sagen Sie, was Sache ist: „Bitte füllen Sie das Feld ‚Vorname‘ aus.“ Ein Hinweis, der nichts erklärt, hilft dem Benutzer nicht weiter. Wenn Sie auf Fehlercodes nicht verzichten können, belassen Sie es in der Fehlermeldung nicht nur bei der Angabe des Codes. Ein Fehlercode ist ein Fehlercode und keine Fehlermeldung. In einer Anwendung schreiben Sie am besten kurz, wo das Problem liegt. In einem abgesetzten Bereich steht, wie der Benutzer es lösen kann und ganz am Schluss ein Fehlercode, wenn nötig.

Wenn es nur darum geht, dass in einem Webformular ein Pflichtfeld vergessen wurde, müssen Sie natürlich keine Romane schreiben. Ein „Bitte füllen Sie das Feld ‚Vorname‘ aus.“ reicht völlig. Aber

Vorbildlich: Kleine Hinweistexte erleichtern das Ausfüllen.

bitte recht freundlich und nicht: „Das Feld ‚Vorname‘ ist auszufüllen!“

**So einfach wie möglich.** Idealerweise akzeptieren Sie Eingaben in allen Formaten: Schreiben Sie nicht vor, wie jemand eine Telefonnummer eingeben soll, sondern nehmen Sie alle Schrägstriche, Bindestriche, Klammern und Pluszeichen entgegen. In der Formularprüfung können Sie alle unnötigen Zeichen ausfiltern und sich die Eingabe so formatieren, wie Sie sie benötigen.

Mitunter brauchen Sie jedoch Daten in einem bestimmten Format, weil es beispielsweise auf eine bestimmte Reihenfolge ankommt. Denken Sie nur an die unterschiedlichen Eingabemöglichkeiten eines Datums: *MM/DD/YY* versus *DD/MM/YY*. Die denkbar schlechteste Vorgehensweise ist es, ein Feld dafür vorzusehen und dem Besucher nicht zu sagen, in welchem Format Sie die Eingabe gerne sehen möchten. Wenn Sie sich dann noch in der Fehlermeldung darüber ausschweigen, gehen Ihre Account-Eröffnungen garantiert in den Keller. Machen Sie stattdessen lieber Folgendes: Teilen Sie die Eingabe über mehrere Felder auf, je eines für den Tag, den Monat und das Jahr. Dies hat außerdem den Vorteil, dass Sie die Eingaben leichter getrennt voneinander prüfen können („35.13.2005“ ist kein sinnvolles Datum). Geben Sie neben dem Eingabefeld auch eine Formatierungshilfe an, indem Sie ein Beispiel aufschreiben, welches Format Sie gerne haben möchten. Die Direktbank hat das auf ihrer Website zur Depotöffnung vorbildlich gelöst (siehe Abbildung): Die JavaScript-Eventhandler *onblur()* und *onfocus()* triggern eine Funktion, die direkt neben dem betreffenden Feld eine kleine Hilfe anzeigt.

Seien Sie aber auch hier nicht über-eifrig: Ein Hinweis beim Verlassen eines Feldes mit *onblur()* ist gut, aber verbinden Sie niemals eine Formularprüfung mit *onblur()* – das geht schnell auf die Nerven.

Wenn sich die möglichen Eingabewerte sowieso nur in einem begrenzten Bereich befinden, ist es noch besser, statt Eingabefeldern Listen zu verwenden. Je eine Liste für 31 Tage und zwölf Monate ist überschaubar. Übertreiben Sie es auch hier nicht: Eine kilometerlange Liste durchzuscrollen nervt. Setzen Sie wichtige Einträge an den Anfang der Liste, beispielsweise die Hauptabnehmerländer. Wie oft habe ich schon „Afghanistan“ gewählt, weil ich zu faul war, zu „Germany“ zu scrollen ... Verwenden Sie Listen oder Radio-Buttons, wenn der Kunde zwischen mehreren Optionen wählen kann, die sich gegenseitig ausschließen. Der Vorteil von Radio-Buttons ist, dass alle Optionen auf einen Blick ersichtlich sind, während Listen schnell unübersichtlich werden. Bei Frageaktionen ist es üblich, Radio-Buttons zu verwenden, da das dem Teilnehmer von echten Umfragen bekannt vorkommt (auf einem Blatt Papier kreuzt er etwas an, und Radio-Buttons ähneln diesem Verhalten). Bei Fragebögen sollten Sie jedoch immer darauf achten, *alle* möglichen Optionen zur Verfügung zu stellen, inklusive „Egal/Weiß nicht“ als vorausgewählten Eintrag. Wenn der Benutzer unter mehreren Optionen wählen darf: In mehrzeiligen Listen können mit der STRG- beziehungsweise BEFEHL-Taste zwar mehrere Einträge selektiert werden, das Wissen darum ist jedoch nicht sehr verbreitet. Für diesen Fall gibt es Checkboxes. Noch ein Tipp zu Umfragen: Geben Sie an, wie viele Seiten insgesamt auszufüllen sind, damit die



Glasklar: Erst beim Klick auf „Bestellung abschicken“ gibt es kein Zurück mehr.

Teilnehmer den zeitlichen Aufwand einschätzen können. Auch hier gilt: Weniger ist mehr: Weniger Fragen/Zeitaufwand = mehr Teilnehmer.

Ein letzter Hinweis zur Gestaltung der Formulare: Falls für ein Eingabefeld nur eine bestimmte Anzahl von Zeichen akzeptiert werden kann, lassen Sie dies den Benutzer erstens wissen und zweitens lassen Sie ihn nicht mehr Zeichen eingeben, als Sie akzeptieren möchten. Für diesen Zweck verwenden Sie in *Input*-Feldern das Attribut *maxlength* wie im Quellcode-Beispiel weiter oben. Falls Sie beispielsweise bei einem E-Mail-Formu-

lar verhindern möchten, dass ein Besucher die Mail an 50 Empfänger schickt, dann weisen Sie den Besucher spätestens bei der Fehlermeldung darauf hin (besser noch vorher, indem Sie unter dem Eingabefeld für die Adressen über die Limitierung informieren).

**Formularprüfung.** Seien Sie bei der Formularprüfung so großzügig wie möglich. Bei einer E-Mail-Adresse gibt es natürlich keine Kompromisse, da jede Mail-Adresse genau ein @-Zeichen enthält, keine Punkte direkt vor oder nach dem @-Zeichen und ansonsten keine Leerzeichen (Sie können natürlich noch mehr Aufwand betreiben um festzustellen, ob die Mail-Adresse gültig ist; in der Regel wird es aber nur darum gehen, Tippfehler abzufangen). Bei einer Telefonnummer oder einer Anschrift gibt es wesentlich mehr Möglichkeiten. Denken Sie auch an Kunden im Ausland, wo Postleitzahlen oft anders formatiert werden und ganze Adressen anders aufgebaut sind (wie etwa

in den USA mit obligatorischer Angabe des Bundesstaates).

Es ist gut, wenn Sie vor der Übermittlung an den Server die Eingaben überprüfen können, also mit JavaScript bei HTML-Formularen und mit ActionScript in Flash. Von dieser ersten Prüfung abgesehen muss die „richtige“ Formularprüfung aber immer auf dem Server passieren – selbst wenn Sie beim Flash Player davon ausgehen, dass der Surfer nicht ActionScript abschalten kann, so wie es bei JavaScript im Browser möglich ist. Den Grund dafür kennen Sie von Fox Mulder aus *Akte X*: „Traue niemandem.“ Eingaben von außen, die durch Cookies, *POST*- oder *GET*-Parameter in das System kommen, sind per Definition nicht vertrauenswürdig, auch wenn Sie die besten Kunden der Welt haben. Wenn Sie dies nicht akzeptieren und entsprechend handeln, ist Ihre Website anfällig für Cross-Site Scripting (XSS) und SQL-Injection. Wie Sie XSS verhindern, entnehmen Sie bitte anderen Artikeln (siehe Literaturliste). Im Zusammenhang mit Formularen sei an dieser Stelle nur kurz auf die damit verbundene Problematik hingewiesen.

**Ab die Post!** Noch etwas, bevor wir zu den Ereignissen kommen, die nach dem Abschicken passieren: Wenn es sich bei Ihrem Formular um einen Bestellvorgang handelt, sollten Sie Ihrem Kunden zeigen, an welcher Stelle im Prozess er sich gerade befindet. Es ist als Kunde ein beruhigendes Gefühl, zu wissen, *wann* eine Bestellung wirksam wird. Im Gegensatz zur Ungewissheit, ob beim nächsten Mausklick schon die Kasse klingelt.

Begehen Sie dabei aber bitte nicht den Fehler, auf „Weiter“-Buttons zu verzichten: Die Anzeige des Checkout-Prozesses ist nur eben das: eine Anzeige. Der nächste Punkt in der Anzeige sollte nicht als Submit-Button missbraucht werden. Es ist nicht verkehrt, sich an ein paar Konventionen zu halten, die sich eingebürgert haben, denn das wird von den Besuchern auch schnell wieder erkannt. Bei Amazon ist dies sehr gut gelöst: Es gibt eine Anzeige über den Fortschritt im Checkout-Prozess und die Schaltfläche, mit der die Bestellung ausgelöst wird, ist nicht mit „Weiter“ beschriftet, sondern unmissverständlich mit „Bestellung abschicken“.

### Listing 3

```
<input type="submit" name="Submit" id="Submit"
value="Anfragen" onclick=
"this.style.visibility='hidden';
document.getElementById
('hinweis').style.visibility = 'visible';" />
<div id="hinweis" style="visibility:hidden;" >
Bitte haben Sie einen Moment Geduld.</div>
```

### More Info

Zum Thema Usability für Websites möchte ich Ihnen drei Bücher empfehlen.

- „Defensive Design for the Web“ von 37signals widmet eines von zehn Kapiteln dem Thema Formulare. Es erschien 2004 im New Riders-Verlag (ISBN: 0-7357-1410-X).
- Einen breiteren Ansatz verfolgt Steve Krug mit seinem „Don't make me think!“, das in deutscher Übersetzung bei mitp erschienen ist. Es gefällt mir sogar noch besser als „Defensive Design“ (ISBN: 3-8266-0890-9).
- Von Thomas Wirth stammt „Missing Links: Über gutes Webdesign“. Ein 330-Seiten-Schmöker, in dem es vor allem um Kommunikation, Wording und Aufmerksamkeit geht. Das Buch erschien im Hanser-Verlag (ISBN: 3-446-22009-7).

- *evolt.org*: „Usable Forms (for an international audience)“: [www.evolt.org/article/Usable\\_Forms\\_for\\_an\\_international\\_audience/4090/15118/](http://www.evolt.org/article/Usable_Forms_for_an_international_audience/4090/15118/)
- Digital Web: „Forms, usability, and the W3C DOM“: [digital-web.com/articles/forms\\_usability\\_and\\_the\\_w3c\\_dom/](http://digital-web.com/articles/forms_usability_and_the_w3c_dom/)
- Dive Into Accessibility: Viele Informationen zum Thema Zugänglichkeit: [diveintoaccessibility.org](http://diveintoaccessibility.org)
- Wikipedia-Eintrag zu Cross-Site Scripting: [de.wikipedia.org/wiki/XSS](http://de.wikipedia.org/wiki/XSS)
- Wikipedia-Eintrag zu SQL-Injection: [de.wikipedia.org/wiki/SQL-Injection](http://de.wikipedia.org/wiki/SQL-Injection)
- FAQ der Newsgroup *de.comp.lang.php* zu Sicherheitsfragen: [www.php-faq.de/ch/ch-security.html](http://www.php-faq.de/ch/ch-security.html)

Ihr Kunde hat das Formular ausgefüllt und es hat auch fehlerfrei die Formularprüfung passiert. Benutzerfreundliche Formulare sind jedoch auch nach dem Submit noch für ihre Besucher da. Sie haben bestimmt schon einmal einen Hinweis in dieser Art gesehen: „Klicken Sie nur einmal auf den ‚Abschicken‘-Button! Wenn Sie doppelt klicken, wird Ihre Kreditkarte zweimal belastet und morgen gibt es schlechtes Wetter!“ Solche Hinweise sind nicht ganz unberechtigt: Die meisten Nutzer klicken alles doppelt an, was sich anklicken lässt, auch wenn ein einfacher Klick völlig ausreichend ist. Ersparen Sie Ihrem Kunden das ungute Gefühl, dass Unfug mit seinem Konto passiert, wenn er doppelt klickt. Direkt nach dem Submit blenden Sie den Button aus, dann kann der Surfer so oft klicken, wie er möchte. Und Sie können den Angst einflößenden Hinweis weglassen. Die

einfachste Möglichkeit war bislang, den Button zu deaktivieren:

```
<input type="submit" name="Submit" id="Submit"
value="Anfragen" onclick="this.disabled=true" />
```

Allerdings ist dieser Ansatz problematisch: Unter Umständen wird der Submit nicht mehr ausgeführt und demnach das Formular gar nicht abgeschickt. Lösung: Blenden Sie den Button aus und einen Hinweis ein (Listing 3).

**Fazit.** Jetzt kann nichts mehr schief gehen: Die Formulardaten sind unterwegs! Die letzte Sache, mit der Sie bei Ihren Besuchern einen positiven Eindruck hinterlassen können, ist, ihnen eine E-Mail mit den Formularinhalten zu schicken. Für Ihre Kunden bedeutet dies, dass die Übermittlung geklappt hat; außerdem haben sie noch eine Kopie für ihre eigenen Unterlagen.

Wenn Sie diese kleinen Tipps umsetzen, werden Sie bessere Formulare bauen als bisher. Ihre Kunden werden mit weniger Frusterlebnissen konfrontiert, und das ist es ja, was wir wollen: Es dem Kunden so einfach wie möglich machen, eine Bestellung abzusetzen.

#### Autor Stefan Freimark

Stefan Freimark ist freier Multimedia-Producer aus Erlangen und hat zwei Laster: gute Filme und die griechische Küche. Hauptsächlich setzt er grafische Konzepte in CSS-Layouts und Flash-Websites um. Wenn dann noch Zeit bleibt, schreibt er an seinem Weblog *my-two-cents.de* weiter.

[stefan@freimark.de](mailto:stefan@freimark.de)  
[www.freimark.de](http://www.freimark.de)

#### Listing 4

Hier sehen Sie den Code eines Affenformulars wie im Text beschrieben. Bitte beachten Sie, dass diese Datei nur die grundlegende Funktionalität zur Verfügung stellt und keine komplette Lösung ist.

```
<!DOCTYPE html PUBLIC "-//W3C//
DTD XHTML 1.0 Transitional//EN"
"DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<title>test</title>
</head>

<body>

<?php
print "von_zeit: " . $_POST["von_zeit"] . "<br />\n";
print "anrede: " . $_POST["anrede"] . "<br />\n";
print "textfeld: " . $_POST["textfeld"] . "<br />\n";
var_dump($_POST["optionen"]);
?>

<form action="<?php print $_SERVER['PHP_SELF'] ?>"
method="post">

<!-- Auswahllisten -->
<?php $liste_von = $_POST["von_zeit"] ?>
<select name="von_zeit" id="von_zeit" size="1">
```

```
<option value="0" <?php if ($liste_von == "0")
{ print "selected=\"selected\""; } ?>
(Bitte ausw&auml;hlen)</option>
<option value="09" <?php if ($liste_von == "09")
{ print "selected=\"selected\""; } ?>
09:00 Uhr</option>
<option value="10" <?php if ($liste_von == "10")
{ print "selected=\"selected\""; } ?>
10:00 Uhr</option>
<option value="11" <?php if ($liste_von == "11")
{ print "selected=\"selected\""; } ?>
11:00 Uhr</option>
</select>

<br />

<!-- Radio-Buttons -->
<label for="anrede_frau">Frau</label><input
name="anrede" id="anrede_frau" type="radio"
value="Frau" <?php if ($_POST["anrede"] ==
"Frau") { print "checked=\"checked\""; } ?> />
<br />
<label for="anrede_herr">Herr</label><input
name="anrede" id="anrede_herr" type="radio"
value="Herr" <?php if ($_POST["anrede"] ==
"Herr") { print "checked=\"checked\""; } ?> />

<br />

<!-- Check-Boxen -->
<label for="zimmer_option1">Seeblick</label>
<input name="optionen[]" id="zimmer_option1"
type="checkbox" value="Seeblick"
```

```
<?php if (in_array("Seeblick",$_POST
["optionen"])) { print "checked=
\"checked\""; } ?> <br />
<label for="zimmer_option2">Klimaanlage</label>
<input name="optionen[]" id="zimmer_option2"
type="checkbox" value="Klimaanlage"
<?php if (in_array("Klimaanlage",
$_POST["optionen"])) { print
"checked=\"checked\""; } ?> />

<br />

<!-- Textarea -->
<textarea name="textfeld" id="textfeld" cols="45"
rows="10"><?php print $_POST ["textfeld"]; ?>
</textarea>

<br />

<!-- Submit -->
<input type="submit" name="Submit" id=
"Submit" value="Anfragen" onclick="this.style.
visibility = 'hidden'; document.getElementById
('hinweis').style.visibility='visible';" />
<div id="hinweis" style="visibility:hidden;" >
Bitte haben Sie einen Moment Geduld.</div>

</form>

</body>
</html>
```