Formulare mit AJAX.. und ohne

== Formulare ==
Formulare sind eine Möglichkeiten um Eingaben des Benutzers zu erhalten. Ein Formular muss mindestens die Attribute "action" und "method" haben.
Das "method"-Attribut gibt die HTTP Request Methode an, über die die Daten transferiert werden. Das kann entweder POST oder GET sein.
GET Daten werden über die URL übertragen - sie sind also für jeden Betrachter und auch in den Logfiles sichtbar. Für Formulareingaben sollte dies niemals verwendet werden.

Die Eingabe wird nicht direkt an das Formular - sondern mit einem HTTP Request an eine zweite URL - geschickt. Diese gibt man über das "action"-Attribut an.
Man kann die Eingabe auch an das Formular selbst schicken - das ist eine gute Praxis, wenn man nicht seinen Code nicht über zu viele Dateien verstreuen will.

Quellcode

```
1. <form action="" method="post">
2. ...
3. </form>
```

== Formularfelder ==
Formulare haben verschiedene Feldtypen wie einzeilige und mehrzeilige Texteingaben, Auswahlboxen und -Listen mit einfach und mehrfach Auswahl und Feldtypen für Dateiuploads.
Die einfachste Platzierung ist schlicht, sehr gängig und enthält eigentlich alle notwendigen Informationen:

Quellcode

```
1. <input type="checkbox" name="checkbox1[]" value="1"/> Beschreibung der Auswahlbox Beschreibung
```

easy-coding.de/Attachment/500/…be06698d0e9402ed7e0179bdc

== Bessere Formularfelder ==
Schönere Formularfelder mit besserer Semantik erhält man, indem die Beschreibung mittels eines Labels dem Eingabefeld zugeordnet wird. Dazu muss dem Eingabefeld eine eindeutige ID vergeben werden:

Quellcode

```
1. <input type="checkbox" id="checkbox1" name="checkbox1[]" value="1"/>
2. <label for="checkbox1">
3. Beschreibung der Auswahlbox
4. </label>
```

Dies hat den praktischen Nebeneffekt, dass man nur den Text klicken braucht um die Auswahlbox zu de-/selektieren.

Eingabefelder lassen sich meist gruppieren. Um so mehr Eingabefelder es gibt, desto eher sollte sich der Entwickler darüber Gedanken machen.
Der Benutzer kann so eine bessere Übersicht gewinnen. Vor allem da HTML hier praktische Elemente liefert, die der Benutzer durch die Verbreitung auf vielen Websites schnell unterbewusst wahrnimmt. Es handelt sich dabei um s.g. "Fieldsets". Um diese "Fieldsets" zu beschriften fügt man eine Legende hinzu.

Quellcode

```
1. <fieldset>
2. <legend>Demo</legend>
3. <input type="checkbox" id="checkbox1" name="checkbox1[]" value="1"/>
```

```
4. <label for="checkbox1">
5. Beschreibung der Auswahlbox
6. </label>
7. </fieldset>
```

easy-coding.de/Attachment/501/…be06698d0e9402ed7e0179bdc

== Verarbeitung in PHP ==
Selbstverständlich gehören alle Elemente die gleichzeitig übertragen werden müssen innerhalb eines einzige Formulars. Sie stehen im Zieldokument unter der Superglobalen $_POST zur Verfügung. Als Index wird der Formularname verwendet. Ein Besonderfall bilden mehrdeutige Elemente. So kann zum Beispiel der Typ Checkbox mehrere Werte annehmen.
Dazu muss man die Array-Schreibweise mit öffnender und schließender eckigen Klammer verwenden und erhält in PHP ein Array, über das man am besten mit einer foreach-Schleife iterierten kann.

Quellcode

```
1. foreach($_POST['checkbox'] as $val) {
2. printf("%s wurde angeklickt<br/>", $val);
3. }
```

Dass $_POST ein normales Array ist erkennt man auch daran, dass sich die print_r Funktion darauf anwenden lässt.

Quellcode

```
1. print_r($_POST);
```

== POST Daten per AJAX übertragen ==
Um Formulardaten über POST mittels AJAX zu übertragen muss ein GET String zusammengebaut werden, so wie man ihn hätte auch direkt im Browser aufrufen können.
Ein vollständiger POST Request sieht Ende wie folgt aus:

Quellcode

```
1. var postData = 'var1=value&var2[]=value&var2[]=value';
2. var req = window.XMLHttpRequest ? new XMLHttpRequest(): new ActiveXObject("Microsoft.XMLHTTP");
3. req.open('POST', 'destination.php', true);
4. req.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
5. req.send(postData);
```

== Formulardaten mit JavaScript sammeln ==
Die selbstgeschriebene Funktion getFormData ist sehr trickreich. Man sagt dass sie das "Formular serialisiert". Der GET String wird zusammengebaut (Alternativ kann sie auch ein Array liefern). Sie iteriert über "form.elements" - also alle Elemente des Formulars. Auswahllisten haben den Sonderfall, dass alle Kindelemente (also die options) abgefragt werden müssen.

An dieser Stelle setze ich die Funktion einfach als gegeben voraus:

Quellcode

```
1. function getFormData(form, asArray) {
2. var ret;
3. var add = function(n, v) {
4. if(asArray) {
5. if(ret == null) ret = new Array();
```

```
6.  ret[n] = escape(v);
7.  } else {
8.  ret = (ret == null ? '' : ret+'&') + n +'='+ escape(v);
9.  }
10. };
11. for(var i=0; i<form.elements.length; i++) {
12. var el = form.elements[i];
13. var type = (el.type || '');
14. if(type.match(/^(text|hidden|textarea)$/i) || (type.match(/^(radio|checkbox)$/i) && el.checked)) {
15. add(el.name, el.value);
16. } else if(el.nodeName.match(/^select$/i)) {
17. for(var j=0; j<el.options.length; j++) {
18. if(el.options[j].selected) {
19. add(el.name, el.options[j].value);
20. }
21. }
22. } else if(el.nodeName.match(/^textarea$/i)) {
23. add(el.name, el.value);
24. }
25. }
26. return ret != null ? ret : (asArray ? new Array() : '');
27. }
```

Alles anzeigen

Die bewährteste Möglichkeit um an Formulardaten zu gelangen, ist es sich in den onsubmit Event des Formular-Objektes einzuhängen.
Der onsubmit Event wird noch ausgeführt bevor das "action"-Attribut ausgelesen wird. Wird das Event mit false terminiert, dann fährt der Browser nicht fort und ignoriert es gar. Das ist die perfekte Voraussetzung für einen AJAX Request.
Als Nebeneffekt wird das onsubmit bei deaktiviertem JavaScript nicht gelesen und man landet direkt bei der Zielseite.

Quellcode

```
1. <form action="post.php" method="post" onsubmit="alert(getFormData(this));return false;">
```

Bei unserem Beispielformular mit nur einem Element sieht der Alert wie folgt aus:
easy-coding.de/Attachment/502/…be06698d0e9402ed7e0179bdc

== Formular per AJAX übertragen ==
Das Übertragen des gesamten Formulars per AJAX ist ein Zusammensetzen der bekannten Funktionen.

Quellcode

```
1. <form action="post.php" method="post" onsubmit="return ajaxPost(this.action,
   getFormData(this),'ajaxResponse');">
```

Der AJAX POST Request wurde in die Funktion ajaxPost übernommen. Diese sieht wie folgt aus:

Quellcode

```
1.  function ajaxPost(url, postData, callback) {
2.  var req;
3.  try {
4.  req = window.XMLHttpRequest ? new XMLHttpRequest(): new ActiveXObject("Microsoft.XMLHTTP");
5.  } catch (e) {
6.  // browser does not have ajax support
7.  }
8.  req.onreadystatechange = typeof callback == 'function' ? callback : function() {
```

```
9.  if (req.readyState == 4 && req.status == 200) {
10. if(typeof callback == 'string') callback = document.getElementById(callback);
11. if(callback) callback.innerHTML = req.responseText;
12. }
13. };
14. req.open('POST', url, true);
15. req.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
16. req.send(postData);
17. return false;
18. }
```

Alles anzeigen

== Best Practice Parallelbetrieb ==
Zum Abschluss möchte ich eine Variante erläutern, wie man einfache Formulare parallel "mit" und "ohne" AJAX betreiben kann.
Ich habe mich dafür entschieden, dass das Formular die Daten an sich selbst schickt. Außerdem will ich unterscheiden ob die Daten per AJAX oder nicht per AJAX gesendet wurden. Daher füge ich allein im onsubmit Event die POST Variable "ajax" hinzu.

Der Formularkopf sieht nach der Bearbeitung wie folgt aus:

Quellcode

```
1. <form
2. action="<?php echo $_SERVER['REQUEST_URI']?>"
3. method="post"
4. onsubmit="return ajaxPost(this.action, 'ajax&'+getFormData(this),'ajaxResponse');">
```

Zusätzlich kümmert man sich im Kopf der Datei um die Bearbeitung des Requests. Falls es sich um einen AJAX Request handelt wird die Meldung direkt ausgegeben und ein exit ausgeführt. Das verhindert, dass der Rest der Seite neu geladen werden muss.
Ansonsten

Quellcode

```
1.  <?php
2.  $response = '';
3.  if(count($_POST)) {
4.  $response = sprintf('<b style="color:green">%s</b>', 'form data ok');
5.  try {
6.  // validation
7.  if(!isset($_POST['radio1'])) throw new Exception('you have to add at least one option');
8.  } catch(Exception $e) {
9.  $response = sprintf('<b style="color:red">%s</b>', $e->getMessage());
10. }
11. if(isset($_POST['ajax'])) {
12. echo $response;
13. exit;
14. }
15. }
16. ?>
```

Alles anzeigen

== Demo ==
Eine Live Demo mit allen möglichen Formularelementen findet ihr unter demo.easy-coding.de/ajax/forms-with-and-without-ajax

. Des weiteren wird der kompletten Code hier als ZIP Archiv zur Verfügung gestellt: download.zip.
easy-coding.de/Attachment/503/…be06698d0e9402ed7e0179bdc