L1_2.1 Formulare mit HTML

1 Erstellen von Formularen mit HTML

Formulare stellen in der Welt der interaktiven und dynamischen Webseiten die Verbindung zum Anwender dar. Der Anwender wird durch sie in die Lage versetzt, Daten einzugeben, und der Server kann auf diese Daten in vielfältiger Weise reagieren.

Typische Beispiele für die Verwendung von Formularen sind Anmeldemasken, Masken zur Durchführung von Banküberweisungen oder zur Reisebuchung, Masken für das Erfassen eines Beitrags in sozialen Netzwerken oder Masken für das Verfassen von E-Mails.

Das form-Element

Ein Formular setzt sich aus dem `<form>`–Tag und den enthaltenen Formularelementen wie einzeilige oder mehrzeilige Textfelder, Checkboxen, Auswahllisten, Radiobuttons etc. zusammen. Meist werden die eingegebenen Daten durch Klick auf einen Button zum Server gesendet.

Sie können an beliebiger Stelle innerhalb des Body-Elements einer HTML-Datei ein Formular definieren.

```
<form action="ziel.php" target="_blank">
verschiedene Formularlemente und HTML-Elemente
</form>
```

Mit `<form>...</form>` definieren Sie ein Formular. Alles, was zwischen dem einleitenden `<form>`-Tag und dem abschließenden Tag `</form>` steht, ist Bestandteil des Formulars. Dabei handelt es sich in der Hauptsache um Elemente wie Texteingabefelder, Auswahllisten oder Schaltflächen.

Im einleitenden `<form>`-Tag bestimmen Sie mit dem Attribut `action`, an welches Zieldokument die Formulardaten beim Absenden des Formulars übertragen werden sollen. Bei uns handelt es sich üblicherweise um eine PHP-Seite. Normalerweise ersetzt die Antwortseite des Servers die aktuelle Seite. Möchte man dies nicht, kann die Antwortseite in einem neuen Tab geöffnet werden. Hierzu ist das `form`-Element durch das optionale Attibut „`target`" mit dem Wert „`_blank`" zu ergänzen.

Das input-Element

Das Form-Tag umschließt alle Formularelemente, hat jedoch keinen Einfluss auf das Aussehen der Seite. Es soll nun ein erstes Formularelement eingefügt werden, ein einzeiliges Texteingabefeld:

```
<input type="text">
```

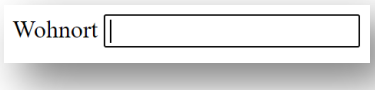
```
<form action="ziel.php">
  <input type="text">
</form>
```

Verschiedene Eingabefelder werden durch den Tag `<input>` eingeleitet und über das Attribut `type` näher spezifiziert. Der Typ "`text`" definiert ein einzeiliges Testeingabefeld. Das `input`-Element wird üblicherweise nicht geschlossen, man spricht auch von einem Standalone-Tag.

Das label-Element

Eine gute Benutzerführung zeichnet sich dadurch aus, dass die Eingabefelder mit einer Beschriftung versehen sind. In HTML steht hierfür das Element `label` zur Verfügung.

```
<label for="ort">Wohnort</label>
<input id ="ort" type="text">
```



Ein `label`-Element gehört immer zu einem Eingabefeld. Die Zuordnung erfolgt über die Attribute `for` beim `label`-Element und `id` beim Eingabefeld. Beide Attribute müssen den gleichen Wert besitzen.

Hinweis: Prinzipiell könnte man auch einfach den gewünschten Text vor das Eingabefeld notieren und auf das `label`-Element verzichten. Das `label`-Element hat den Vorteil, dass es logisch mit dem Eingabefeld verbunden ist. Klickt man auf die Beschriftung, erhält das zugehörige Feld den Fokus. Dies erhöht die Bedienerfreundlichkeit besonders dann, wenn das Eingabefeld sehr klein ist, beispielsweise eine Checkbox. Bei Sprachausgaben (Screenreader) wird die Beschriftung automatisch vorgelesen, wenn das zugehörige Eingabefeld den Fokus erhält.

Das fieldset- und das legend-Element

Mehrere inhaltlich zusammengehörende Eingabeelemente werden häufig zu einem Block zusammengefasst. Das fieldset-Element spannt einen Rahmen um alle enthaltenden Eingabeelemente auf.

```
<fieldset>
     <label for="str">Straße</label>
     <input id ="str" type="text"><br>

     <label for="ort">Wohnort</label>
     <input id ="ort" type="text">
</fieldset>
```

Ein solcher Block kann zusätzlich mit einer Beschriftung versehen werden. Dazu dient das `legend`-Element.

```
<fieldset>
     <legend>Adresse</legend>
     <label for="str">Straße</label>
     <input id ="str" type="text"><br>

     <label for="ort">Wohnort</label>
     <input id ="ort" type="text">
</fieldset>
```

Das input-Element als Schaltfläche

In der Regel besitzen Formulare eine Schaltfläche, welche das Versenden der Formulardaten startet. Nach Klick auf die Schaltfläche sammelt der Browser alle zum Formular gehörenden Daten ein und sendet diese in der URL codiert an das im `action`-Attribut hinterlegte Zieldokument. Eine alternative Codierungsart wird später vorgestellt.

Schaltflächen für das Absenden von Daten werden ebenfalls über das bereits bekannte `input`-Element erstellt. Eine separate Beschriftung mit Hilfe des `label`-Tags ist hier nicht sinnvoll.

```
<input type="submit" value="Daten senden">
```

Der Type „`submit`" macht das `input`-Element zu einer Schaltfläche. Die Aufschrift kann über das Attribut value festgelegt werden.

Das Formular sieht nun wie folgt aus:

```
<form action="ziel.php">
     <fieldset>
          <legend>Adresse</legend>

          <label for="str">Straße</label>
          <input id ="str" type="text">

          <label for="ort">Wohnort</label>
          <input id ="ort" type="text">

          <input type="submit" value="Daten senden">
     </fieldset>
</form>
```

2. Kodierung der Daten

Sehr häufig enthalten Formulare mehrere Eingabefelder. Um diese später eindeutig identifizieren zu können, muss jedem Eingabefeld – außer den Schaltflächen – ein `name`-Attribut hinzugefügt werden. Auf die Darstellung des Formulars im Browserfenster hat dies keinerlei Auswirkung. Der nun fertige HTML-Code für das Formular sieht dann wie folgt aus.

```
<form action="ziel.php">
     <fieldset>
          <legend>Adresse</legend>

          <label for="str">Straße</label>
          <input id ="str" name="st" type="text">

          <label for="ort">Wohnort</label>
          <input id ="ort" name="or" type="text">

          <input type="submit" value="Daten senden">
     </fieldset>
</form>
```

Nehmen wir an, der Besucher der Seite füllt das Formular wie oben dargestellt aus. Was passiert nun, wenn das Formular ausgelöst wird, der Besucher der Seite also auf den „Sende"-Knopf klickt.

Der Browser erstellt eine URL, die auf das Zieldokument zeigt und die Formulardaten in Form einer Query verpackt. Anschließend wird eine Anfrage an diese URL gestartet.

1. Der Browser ermittelt zunächst den Wert des `action`-Attributs des `form`-Elements. Der Wert dieses Attributs bildet den ersten Teil der zu erzeugenden URL. Wird im action-Attribut kein Protokoll angegeben, handelt es sich um eine relative Adressierung, d.h. der Pfad wird ausgehend vom Ort der aktuellen Datei ermittelt. Im konkreten Fall zeigt das `action`-Attribut auf eine Datei namens „ziel.php", welche sich im gleichen Verzeichnis befindet, wie die HMTL-Seite, welche das Formular erzeugt.

   Unter der Annahme, dass die HTML-Seite mit dem Formular unter der Adresse http://localhost/fikitv/v1/formular.html erreichbar ist, dann lautet der erste Teil der URL

   http://localhost/fikitv/v1/**ziel.php**

2. Im zweiten Schritt werden alle Eingabefelder analysiert, die über ein `name`-Attribut verfügen. Die Daten werden als Query-Komponente angehängt. Dabei bildet ein Eingabefeld einen Parameter, wobei der Parametername dem Namen des Feldes entspricht (`name`-Attribut) und der Parameterwert der Eingabe des Benutzers. Ausgehend vom obigen Beispiel wird folgende Query-Komponente gebildet:

   st=Königsstraße&or=Stuttgart

   Beachten Sie, dass die beiden Parameter durch ein Et-Zeichen getrennt werden.

Aus beiden Teilergebnissen kann nun die resultierende URL erzeugt werden. Die Query-Komponente wird dabei durch ein Fragezeichen vom Zieldokument getrennt:

http://localhost/fikitv/v1/ziel.php**?**st=Königsstraße&or=Stuttgart

Zieldokument — Query-Komponente

Im letzten Schritt führt der Browser eine Anfrage mit der soeben erzeugte URL aus, die Antwort des Webservers wird anschließend im Browserfenster angezeigt.