



CODERS.BAY

# **Formulare**

## **Schrödinger Kapitel 06**

- `<form action="forms/hallowelt.php" method="post" accept-charset="utf-8">`
- wird benötigt um Daten zum Server zu schicken
  - ▶ `action="forms/hallowelt.php"`
    - ▶ action-Attribut gibt an wohin mit den Daten
  - ▶ `method="get"`
    - ▶ wie die Daten an den Server geschickt werden
  - ▶ `accept-charset="utf-8"`
    - ▶ Character Encoding, mit dem die Daten zum Server geschickt werden sollen

- ▶ ein einfaches Eingabefeld für Text
- ▶ `type="text"`
  - ▶ gibt an wie die Daten eingegeben werden, in dem Fall als Text
- ▶ `name="inputtext"`
  - ▶ Der Name des Eingabefeldes ist auch der Name, unter dem der Server diese Daten finden wird
- ▶ `type="password"`
  - ▶ verschleierte Eingabe
- ▶ `max-length`
  - ▶ maximale Zeicheneingabemöglichkeit



## BUTTON



- ▶ `<button type="submit">Abschicken</button>`
  - ▶ Schickt die Daten des Formular ab
- ▶ `type="reset"`
  - ▶ Formular wird komplett geleert
- ▶ `type="button"`
  - ▶ hat kein eigenes Verhalten



## LABEL



Label markiert, dass der umschlossene Text die Beschriftung des Eingabefeldes ist. Dazu muss das **for**-Attribut des **<label>**s mit der **id** des Eingabefeldes übereinstimmen. Das **<label>**-Tag kann "sein" Eingabefeld auch umschließen, dann entfällt das **for**-Attribut.

```
<form action="http://localhost:8080/forms/hallowelt.php" method="post">  
  <label for="testeintrag">Wie heißt du?</label>  
  <input type="text" name="vorname" id="testeintrag">  
  <button>Sag mal hallo!</button>  
</form>
```



## PLACEHOLDER



- ▶ `<input name="meinEintrag" placeholder="hier müssen sie was rein schreiben">`
  - ▶ Platzhaltertext
  - ▶ verschwindet sobald das Feld beschrieben wird
- ▶ CSS
  - ▶ `::placeholder { color: red; }`
  - ▶ `::-webkit-input-placeholder { color: red; }`
  - ▶ `::-ms-input-placeholder { color: red; }`

# GET UND POST

- ▶ **get**-Request
  - ▶ holen Seiten
  - ▶ Parameter, die mit dem get-Request geschickt werden, sagen dem Server, welche Daten man haben möchte
- ▶ **post**-Request
  - ▶ Daten an den Server zu schicken, die der dann speichern soll

## RADIOBUTTONS

Männlein ☒ Weiblein ☐

```
<label for="geschlecht_1m">Männlein</label>
<input type="radio" name="geschlecht" value="m" id="geschlecht_1m" checked="checked">
<label for="geschlecht_1w">Weiblein</label>
<input type="radio" name="geschlecht" value="w" id="geschlecht_1w">
```

- ▶ **for** verheirachtet das Label mit dem Input
- ▶ **checked** macht eine Vorauswahl für den Benutzer

```
<label>Männlein</label>
<input type="radio" name="geschlecht" value="m">
<label>Weiblein</label>
<input type="radio" name="geschlecht" value="w">
```



## CHECKBOXEN

Männlein ☒ Weiblein ☐

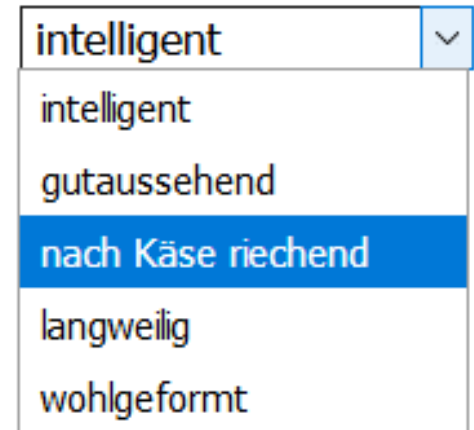
```
<label for="geschlecht_m">Männlein</label>
<input type="checkbox" name="geschlecht" value="m" id="geschlecht_m" checked="checked">
<label for="geschlecht_w">Weiblein</label>
<input type="checkbox" name="geschlecht" value="w" id="geschlecht_w">
```

- ▶ **for** verheiratet das Label mit dem Input
- ▶ **checked** macht eine Vorauswahl für den Benutzer
- ▶ bei Checkboxes sind mehrfach Auswahl möglich

```
<label>Männlein</label>
<input type="checkbox" name="geschlecht" value="m">
<label>Weiblein</label>
<input type="checkbox" name="geschlecht" value="w">
```

# SELECT-BOXEN

```
<select name="zusatz">
  <option value="intelligent">intelligent</option>
  <option value="gutaussehend">gutaussehend</option>
  <option value="nach Käse riechend">nach Käse riechend</option>
  <option value="langweilig">langweilig</option>
  <option value="wohlgeformt">wohlgeformt</option>
</select>
```



intelligent

intelligent

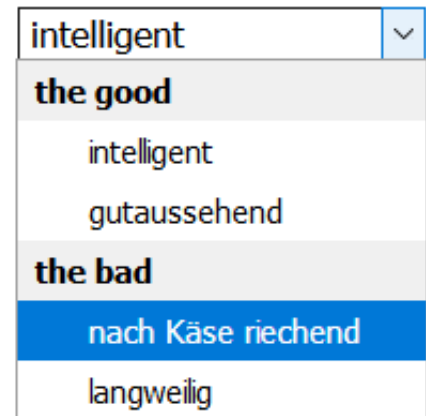
gutaussehend

**nach Käse riechend**

langweilig

wohlgeformt

```
<select name="zusatz">
  <optgroup label="the good">
    <option value="intelligent">intelligent</option>
    <option value="gutaussehend">gutaussehend</option>
  </optgroup>
  <optgroup label="the bad">
    <option value="nach Käse riechend">nach Käse riechend</option>
    <option value="langweilig">langweilig</option>
  </optgroup>
</select>
```



intelligent

**the good**

intelligent

gutaussehend

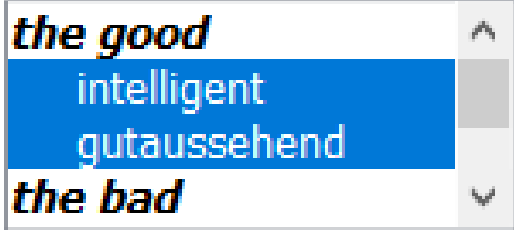
**the bad**

**nach Käse riechend**

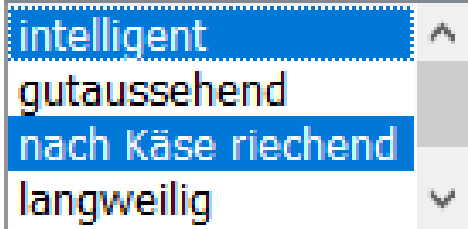
langweilig

# SELECT-BOXEN MIT MEHRFACH AUSWAHL

```
<select name="zusatz" multiple="multiple">
  <option value="intelligent">intelligent</option>
  <option value="gutaussehend">gutaussehend</option>
  <option value="nach Käse riechend">nach Käse riechend</option>
  <option value="langweilig">langweilig</option>
  <option value="wohlgeformt">wohlgeformt</option>
</select>
```



```
<select name="zusatz" multiple="multiple">
  <optgroup label="the good">
    <option value="intelligent">intelligent</option>
    <option value="gutaussehend">gutaussehend</option>
  </optgroup>
  <optgroup label="the bad">
    <option value="nach Käse riechend">nach Käse riechend</option>
    <option value="langweilig">langweilig</option>
  </optgroup>
</select>
```





- ▶ gibt eine Liste von Einträgen an, die dem Benutzer vorgeschlagen werden, wenn er in ein Texteingabefeld tippt
- ▶ halbe Select-Box: Es werden mögliche Werte vorgeschlagen, aber der Benutzer kann auch eigene Eingaben machen

```
<input type="text" list="adjektive" name="zusatz[]">  
<datalist id="adjektive">  
  <option value="intelligent">  
  <option value="gutaussehend">  
  ...
```



## VERSTECKTE FELDER



- ▶ Daten mit Formularen mitschicken, die der User nicht sehen soll

```
<input type="hidden">
```



## TEXTAREA



```
<textarea name="kommentar" cols="50" rows="5">Der Wert der Textarea</textarea>
```

- ▶ Eingabevalidierung heißt die Prüfung, ob die Eingaben des Benutzers korrekt sind. Steht im E-Mail-Feld wirklich eine E-Mail-Adresse? Und ein Datum im Datumsfeld? Und sind alle Felder ausgefüllt, die der Server dringend braucht?

Typ	IE 11	Edge	Firefox	Chrome	Safari	Beschreibung
type="email"	ja	ja	ja	ja	ja	Lässt nur gültige E-Mail-Adressen zu.
type="url"	ja	ja	ja	ja	ja	Lässt nur gültige <b>absolute</b> URLs zu.
type="datetime-local"	nein	ja	nein	ja	nein	Eingabe von Datum und Uhrzeit. Der Namensteil <code>local</code> bezieht sich darauf, dass keine Zeitzone zur Angabe gehört, die Zeit wird immer in der lokalen Zeitzone interpretiert.
type="date"	nein	ja	ja	ja	nein	Eingabe eines Datums. Chrome bietet hier schon einen schicken Kalender zur Auswahl an.
type="number"	ja	ja	ja	ja	ja	Lässt nur Zahlen zu und hat schicke Knöpfe zum Hoch- und Runterzählen.
type="color"	nein	ja	ja	ja	nein	Farbauswahl. Die haben echt an alles gedacht. Chrome verdient sich hier mal wieder ein Fleißsternchen mit dem Auswahldialog.
type="search"	ja	ja	ja	ja	ja	Unterscheidet sich in der Funktion nicht von <code>type="text"</code> , kann aber vom Browser dargestellt werden wie ein Suchfeld des Betriebssystems. Das kann sehr unscheinbar sein, bei mir fügen zum Beispiel Chrome und IE ein X im Feld hinzu, um es mit einem Klick zu leeren.
pattern="..."	ja	ja	ja	ja	ja	Das <code>pattern</code> -Attribut kann bei Textfeldern (und einigen anderen) verwendet werden, um ein Muster anzugeben, zu dem die Eingabe passen muss. Die Erklärung kommt gleich.
required	ja	ja	ja	ja	ja	Das Attribut <code>required</code> markiert ein Feld als Pflichtfeld: Wenn es nicht ausgefüllt ist, sieht man eine Fehlermeldung.



```
<style>
input { color: black; }
input[pattern] { border: 2px solid #bbbbff; }
input[type=email] { border: 2px solid #bbffff; }
</style>

<input type="text" name="text1">
<input type="email" name="email">
<input type="text" name="pattern" pattern="[a-z]">
```





## FILE UPLOAD



- ▶ Das Formular muss mit `method="post"` geschickt werden, ein `get`-Request kann keine Dateien transportieren. Da der Server aber meistens die Datei speichern soll, passt das schon.
- ▶ Am Formular muss das Attribut `enctype="multipart/form-data"` gesetzt werden. Dieses Attribut steuert, wie Formulardaten im `post`-Request verpackt werden, und wenn Dateien im Spiel sind, muss es dieser Wert sein. Ansonsten gibt es keinen Grund, das Attribut zu setzen

```
<form method="post" action="upload.php" enctype="multipart/form-data">  
  Datei: <input name="datei" type="file">  
</form>
```

## ➤ Aufgabe 1

- ▶ Schrödinger lernt HTML, CSS und JS  
Kapitel 6 durcharbeiten (Beispiele machen, ggf. in Aufgabe 5 einfließen lassen)

## ➤ Aufgabe 2

Arbeite bei deiner Webseite zum Thema Reisen weiter  
Benutze HTML5 Elemente für eine korrekte semantische Auszeichnung

Die Seiten sollen beinhalten:

- ▶ Titel (steht dein Name)
- ▶ Formular mit allen Elementen die du kennen gelernt hast  
Achte auf die Verbindung von label und for

Benutze für das Styling nur noch dein externes Stylesheet.  
Kein Inlinestyling mehr!!

Stylinganweisung die in **verschiedenen Kombinationen** verwendet werden müssen:

- ▶ Style das Formular!

## ➤ Aufgabe 3

- ▶ weiterführende Links lesen und in Aufgabe 5 umsetzen

## Abgabemodalität:

Erstelle einen Ordner mit deinem Namen

In diesem Ordner erstelle einen zweiten Ordner mit der Bezeichnung „Aufgabe 05“.

Die .css Datei wird in einem eigenen Ordner namens styling im abzugebenden Ordner gespeichert.

Die Bilder werden in einem eigenen Ordner namens img im abzugebenden Ordner gespeichert.

In dem Ordner „Aufgabe 05“ sind alle Datei, Ordner und alle Bilder die du benutzt, abgespeichert.

Die Aufgabe wird Freitags mit einem USB-Stick eingesammelt. Auf dem USB-Stick kopierst du den Ordner mit deinem Namen.

Überprüfe die Datei am Stick ob alle Verlinkungen ordnungsgemäß funktionieren!



CODERS.BAY

ENDE