KAPITEL 4

Formulare

Formulare werden umfassend unterstützt. Viele Komponenten dienen vor allem dazu, ansprechende Formulare zu gestalten, die mit jeder Bildschirmbreite gut zurechtkommen.

Struktur eines Formulars

Formularelemente erhalten automatisch die richtige Formatierung. Als Container für Steuerelemente wird die Klasse `.form-control` benutzt. Elemente, die eine steuerbare horizontale Ausdehnung haben, also `<input>`, `<textarea>` und `<select>`, werden auf eine Breite von 100 % des übergeordneten Containers gesetzt. Mit `.form-group` werden Gruppen aus Label und Eingabefeld gebildet, die sich je nach verfügbarer Breite automatisch nebeneinander oder übereinander anordnen.

Einfache Formularelemente

Hier ein Beispiel für ein typisches Formular:

Beispiel 4-1: Standardaufbau eines Formulars (Form_Base.html)

```
<form>
  <div class="form-group">
    <label for="txtMail">E-Mail</label>
    <input type="email" class="form-control"
           id="txtMail" placeholder="E-Mail">
  </div>
  <div class="form-group">
    <label for="txtPassword">Kennwort</label>
    <input type="password" class="form-control"
           id="txtPassword" placeholder="Password">
```

```
    </div>
    <div class="form-group">
      <label for="txtFile">File selection</label>
      <input type="file" id="txtFile">
      <p class="form-text small">Dies ist die Hilfe.</p>
    </div>
    <div class="checkbox">
      <label>
          <input type="checkbox"> Speichern
      </label>
    </div>
    <button type="submit" class="btn btn-secondary">Senden</button>
  </form>
```

Der äußere Teil ist immer die Elementgruppe `.form-group`. Das Element wird nochmals mit `.form-control` ausgezeichnet. Alle anderen Teile benötigen keine Klassen.

Abbildung 4-1: Ein einfaches Formular

Eingabegruppen

Neben einfachen Elementen können auch komplexere Elemente aus einfacheren zusammengesetzt werden. Dies wird dann als Eingabegruppe bezeichnet. Elementgruppe und Eingabegruppe dürfen nicht gemischt eingesetzt werden, stattdessen werden Eingabegruppen als Kindelement verschachtelt.

Einzeilige Formulare

Einzeilige Formulare stehen ab einer Breite von 768 Pixeln zur Verfügung. Mit »einzeilig« ist gemeint, dass der Feldname (Label), das Eingabefeld und weitere Elemente nebeneinander stehen können, solange der horizontale Platz ausreicht. Sie werden mit `.form-inline` eingeleitet. Das umschließende `<form>`-Tag ist optional. Es kann von der Logik der Seite oder dem Browserverhalten her benötigt werden – Bootstrap reagiert darauf jedoch nicht. Der Standardwert für die Breite der Elemente mit variabler Ausdehnung ist »auto«. Die Breite wird also innerhalb des umschließenden Containers optimiert. Es kann notwendig sein, die Breite individuell zu steuern.

Beispiel 4-2: Kompaktes Formular (Form_Inline.html)

```
<form class="form-inline">
  <div class="form-group">
    <label for="exampleInputName2">Name</label>
    <input type="text" class="form-control"
           id="exampleInputName2" placeholder="Der Name">
  </div>
  <div class="form-group">
    <label for="exampleInputEmail2">E-Mail</label>
    <input type="email" class="form-control"
           id="exampleInputEmail2"
           placeholder="name@email.com">
  </div>
  <div class="w-100 p-4">
    <button type="submit"
            class="btn btn-secondary float-right">
      Senden
    </button>
  </div>
</form>
```

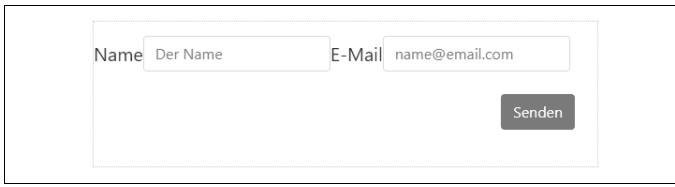

Abbildung 4-2: Ein Formular mit horizontaler Ausrichtung (breit)

Dasselbe Formular sieht bei geringer Bildschirmbreite folgendermaßen aus:

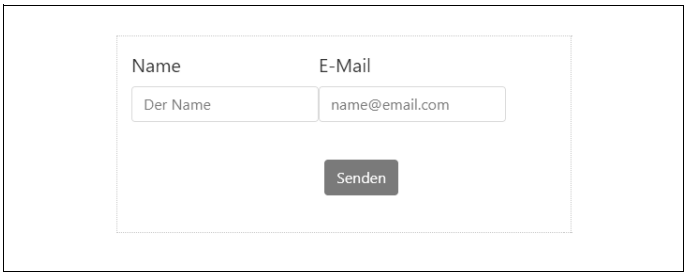

Abbildung 4-3: Ein Formular mit horizontaler Ausrichtung (schmal)

ARIA-Label benutzen

Sie sollten auch bei einzeiligen Formularen immer Label einsetzen. Screenreader können sonst keine sinnvolle Sprachausgabe erzeugen. Nutzen Sie `.sr-only`, damit die Label auf normalen Ausgabegeräten nicht angezeigt werden. Wenn eine barrierefreie Ausgabe erforderlich ist, sollten darüber hinaus immer die vom Standard geforderten Attribute `aria-label` oder `aria-labelledby` genutzt werden. Während `aria-label` den Bezeichnungstext direkt enthält, verweist `aria-labelledby="id"` auf die ID eines anderen Elements auf der Seite, das den Bezeichnungstext liefert.

Das folgende Beispiel zeigt den Einsatz von unsichtbaren Bezeichnungstexten. Für normale Benutzer wird das Wasserzeichen (`placeholder`) benutzt, Screenreader sehen dagegen das Label.

Beispiel 4-3: Formular mit Wasserzeichen und für Screenreader

```
<form class="form-inline">
    <div class="form-group">
    <label class="sr-only" for="exampleInputEmail3">E-Mail</label>
    <input type="email" class="form-control"
           id="exampleInputEmail3" placeholder="E-Mail">
    </div>
```

```
    <div class="form-group">
    <label class="sr-only"
           for="exampleInputPassword3">Kennwort</label>
    <input type="password" class="form-control"
           id="exampleInputPassword3" placeholder="Kennwort">
    </div>
    <div class="checkbox">
    <label>
        <input type="checkbox"> Merken
    </label>
    </div>
    <button type="submit" class="btn btn-secondary">Anmelden</button>
</form>
```

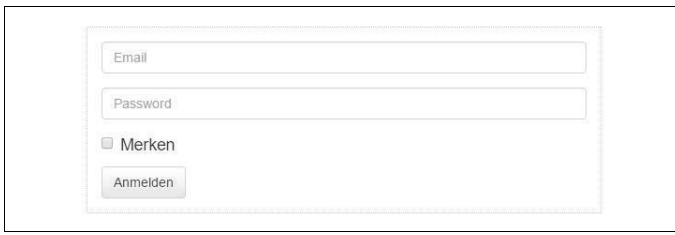

Abbildung 4-4: Ein Formular ohne Label

Formularelemente mit Bausteinen

Eingabefelder, speziell solche für die Eingabe von Text, können mit Text, Symbolen oder Schaltflächen links oder rechts ergänzt werden. Das ist besonders interessant, wenn Werte erfasst werden, die Maßeinheiten haben. Benutzen Sie die Klasse `.input-group` mit `.input-group-prepend` oder `.input-group-append`, um Anzeigeelemente vor oder nach `.form-control` zu erstellen. Das funktioniert sehr gut mit allen `<input>`-Elementen, jedoch nur eingeschränkt mit `<select>` und kaum mit `<textarea>`.

Tooltipps und überlagernde Effekte wie z.B. Modals oder Popover erfordern darüber hinaus weiteren Aufwand. Zumindest ist die Option `container: 'body'` in JavaScript erforderlich, um Seiteneffekte zu vermeiden. Der Parameter bestimmt, wo das dynamische Element im DOM (Document Object Model) der Seite eingefügt wird.

```
$("#toolbarBtn1").dropdown({
    container: 'body';
});
```

Die Spaltenklassen des Rasters lassen sich nicht mit Eingabegruppen mischen. Stattdessen sollte bei Bedarf die gesamte Eingabegruppe in einen Container platziert werden, der seinerseits mit Rastermaßen versehen wird.

Label sind immer sinnvoll. Auch wenn beim konkreten Layout kein Bedarf besteht, werden damit die Screenreader unterstützt. Hier ist wieder der Einsatz von `.sr-only` angebracht.

Beispiel 4-4: Eingabefelder (Toolbar_Inputgroups.html)

```
<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text" id="basic-addon1">@</span>
  </div>
  <input type="text" class="form-control"
         placeholder="User"
         aria-describedby="basic-addon1">
</div>

<div class="input-group">
  <input type="text" class="form-control"
         placeholder="Email"
         aria-describedby="basic-addon2">
  <div class="input-group-append">
    <span class="input-group-text" id="basic-addon2">
      @example.de
    </span>
  </div>
</div>

<div class="input-group">
  <div class="input-group-prepend">
     <span class="input-group-text">&euro;</span>
  </div>
  <input type="text" class="form-control"
         aria-label="Betrag (in EUR)">
  <div class="input-group-append">
     <span class="input-group-text">.00</span>
  </div>
</div>
```

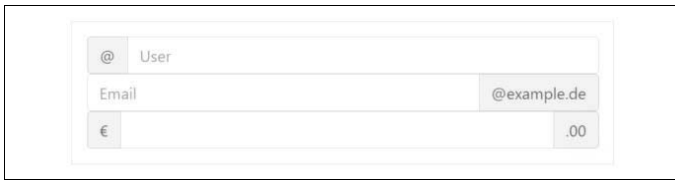

Abbildung 4-5: Eingabefelder

Größen

Die Größen lassen sich relativ in den üblichen vier Stufen festlegen. Dies passiert auf der Gruppe, sodass einzelne Eingabefelder nicht immer wieder mit der Klasse ausgestattet werden müssen.

Beispiel 4-5: Größen der Eingabefelder (Toolbar_InputSize.html)

```
<div class="input-group input-group-lg">
  <div class="input-group-prepend">
    <span class="input-group-text" id="sizing-addon1">@</span>
  </div>
  <input type="text" class="form-control form-control-lg"
         placeholder="Username"
         aria-describedby="sizing-addon1">
</div>

<div class="input-group">
  <div class="input-group-prepend">
    <span class="input-group-text" id="sizing-addon2">@</span>
  </div>
  <input type="text" class="form-control"
         placeholder="Username"
         aria-describedby="sizing-addon2">
</div>

<div class="input-group input-group-sm">
  <div class="input-group-prepend">
    <span class="input-group-text" id="sizing-addon3">@</span>
  </div>
  <input type="text" class="form-control form-control-sm"
         placeholder="Username"
         aria-describedby="sizing-addon3">
</div>
```

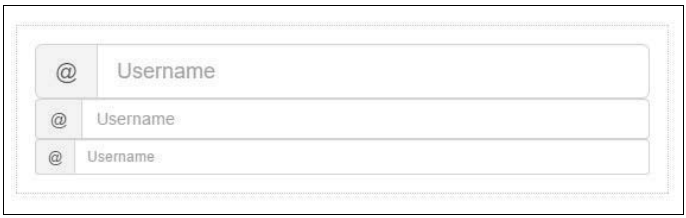

Abbildung 4-6: Größen der Eingabefelder

Umgang mit Kontrollkästchen und Optionsfeldern

Kontrollkästchen und Optionsfelder müssen manchmal direkt neben Eingabefeldern platziert werden. Erneut werden Gruppen verwendet, um die Zuordnung vorzunehmen.

Beispiel 4-6: Eingabefelder mit Optionen (Toolbar_Inputradio.html)

```
<div class="row">
<div class="col-lg-6">
  <div class="input-group">
    <div class="input-group-prepend">
      <span class="input-group-text">
        <input type="checkbox" aria-label="">
      </span>
    </div>
    <input type="text" class="form-control" aria-label="">
  </div>
</div>
<div class="col-lg-6">
  <div class="input-group">
    <div class="input-group-prepend">
      <span class="input-group-text">
        <input type="radio" aria-label="">
      </span>
    </div>
    <input type="text" class="form-control" aria-label="">
  </div>
</div>
</div>
```

Für eine sinnvolle Funktion ist hier unbedingt JavaScript erforderlich – beispielsweise um das zugeordnete Eingabefeld zu aktivieren.

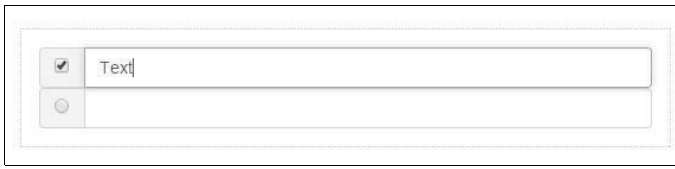

Abbildung 4-7: Eingabefelder mit Optionen

Ergänzende Schaltflächen

Schaltflächen in Eingabefeldgruppen werden ebenso behandelt. Auch hier wird `.input-group-append` oder `.input-group-prepend` benutzt. Allerdings fällt das innere `<span>` weg.

Beispiel 4-7: Eingabefelder mit Schaltflächen (Toolbar_InputBtn.html)

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <span class="input-group-prepend">
        <button class="btn btn-secondary" type="button">
          Los!
        </button>
      </span>
      <input type="text" class="form-control"
             placeholder="Suche nach ...">
    </div>
  </div>
  <div class="col-lg-6">
    <div class="input-group">
      <input type="text" class="form-control"
             placeholder="Suche nach...">
      <span class="input-group-append">
        <button class="btn btn-secondary" type="button">
          Los!
        </button>
      </span>
    </div>
  </div>
</div>
```

Abbildung 4-8: Eingabefelder mit Schaltflächen

Die Schaltflächen lassen sich wiederum mit Aufklappmenüs versehen:

Beispiel 4-8: Eingabefelder mit Menüs (Toolbar_DropDownGroup.html)

```
<div class="row">
  <div class="col-lg-6">
    <div class="input-group">
      <div class="input-group-prepend">
        <button type="button"
                class="btn btn-secondary dropdown-toggle"
                data-toggle="dropdown" aria-haspopup="true"
                aria-expanded="false">
          Auswahl
        </button>
        <div class="dropdown-menu">
          <a class="dropdown-item" href="#">Details</a>
          <a class="dropdown-item" href="#">Kopieren</a>
          <a class="dropdown-item" href="#">Verschieben</a>
          <div role="separator" class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Löschen</a>
        </div>
      </div>
      <input type="text" class="form-control" aria-label="...">
    </div>
  </div>
  <div class="col-lg-6">
    <div class="input-group">
      <input type="text" class="form-control" aria-label="...">
      <div class="input-group-append">
        <button type="button"
                class="btn btn-secondary dropdown-toggle"
                data-toggle="dropdown" aria-haspopup="true"
                aria-expanded="false">
          Auswahl
        </button>
```

```
        <div class="dropdown-menu dropdown-menu-right">
          <a class="dropdown-item" href="#">Details</a>
          <a class="dropdown-item" href="#">Kopieren</a>
          <a class="dropdown-item" href="#">Verschieben</a>
          <div role="separator" class="dropdown-divider"></div>
          <a class="dropdown-item" href="#">Löschen</a>
        </div>
      </div>
    </div>
  </div>
</div>
```

Abbildung 4-9: Eingabefelder mit Menüs

Für die Aufklappmenüs wird *Popper* benötigt:

```
<script src="../node_modules/popper.js/
dist/umd/popper.js"></script>
```

Beachten Sie hier, dass ohne Packer das UMD-Bundle benutzt werden muss. Dieses Skript muss *vor* Bootstrap geladen werden.

Ebenso ist die Unterteilung einer Schaltfläche in Segmente möglich:

Beispiel 4-9: Eingabefelder mit Segmentschaltflächen (Toolbar_InputBtnForm.html)

```
<div class="input-group">
  <div class="input-group-prepend">
    <button class="btn btn-success">Freigeben</button>
  </div>
  <input type="text" class="form-control" aria-label="...">
</div>
```

```
<div class="input-group">
  <input type="text" class="form-control" aria-label="...">
  <div class="input-group-append">
    <button class="btn btn-warning">Gesperrt</button>
  </div>
</div>
```

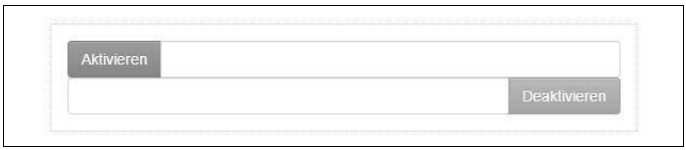

Abbildung 4-10: Eingabefelder mit Segmentschaltflächen

Das folgende Formularelement besteht aus drei Bausteinen: einem Euro-Symbol, dem Eingabefeld und der Angabe ».00« als vordefiniertem Nachkommawert:

Beispiel 4-10: Formular mit Bausteinen (Form_MultiInput.html)

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="exampleInputAmount">
       Betrag (in EUR)
    </label>
    <div class="input-group">
      <div class="input-group-prepend">
        <span class="input-group-text">$</span>
      </div>
      <input type="text" class="form-control"
             id="exampleInputAmount" placeholder="Betrag">
      <div class="input-group-append">
            <span class="input-group-text">.00</span>
          </div>
      </div>
    </div>
  </div>
  <button type="submit" class="btn btn-primary">Überweisen</button>
</form>
```

Hier wird die Klasse `.input-group` genutzt. Das Label (Zeilen 3 bis 5) wird nur Screenreadern angezeigt und bleibt ansonsten unsichtbar.

Die »Extras« vor und hinter dem Element werden mit `.input-group-addon` dekoriert.

Abbildung 4-11: Ein Formular ohne Label

Horizontale Formulare

Um Label und Felder zu platzieren, wird das Raster eingesetzt. Damit die Umbrüche unterdrückt werden, die normalerweise einen kompletten Block aus Beschriftung (Label) und Feld von der nächsten Gruppe trennen, wird `.row` benutzt. Die Angabe der Klasse erfolgt in einem Containerelement, das entweder das umschließende `<form>`-Element oder ein äquivalent eingesetztes `<div>` ist. Die ohnehin erforderliche Gruppierung mit `.form-groups` führt dazu, dass sich die Gruppe wie eine Reihe verhält. Sie können `.row` trotzdem einsetzen, weil sich einige Editoren sonst über das Fehlen der Klasse beschweren – sichtbare Effekte werden dennoch nicht erzielt.

Beispiel 4-11: Horizontales Formular (Form_Horizontal.html)

```
<form class="row">
  <div class="form-group">
    <label for="inputEmail3"
           class="col-sm-2 form-control-label">eMail</label>
    <div class="col-sm-10">
        <input type="email" class="form-control"
               id="inputEmail3" placeholder="E-Mail">
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword3" class="col-sm-2
        form-control-label">Kennwort</label>
    <div class="col-sm-10">
        <input type="password" class="form-control"
               id="inputPassword3" placeholder="Kennwort">
    </div>
  </div>
```

```
    <div class="form-group">
      <div class="offset-sm-2 col-sm-10">
          <div class="checkbox">
          <label>
              <input type="checkbox"> Zugang merken
          </label>
          </div>
      </div>
    </div>
    <div class="form-group">
      <div class="offset-sm-2 col-sm-10">
          <button type="submit" class="btn btn-secondary">
          Anmelden</button>
      </div>
    </div>
  </form>
```

Hier stehen die Label links neben den Elementen. Bei geringer Breite fällt dieses Layout (im Bild ist die Breite > 768 Pixel) wieder auf das vorher gezeigte Schema mit darüber platzierten Labeln zurück.

Abbildung 4-12: Ein Formular mit Label neben dem Feld

Eingabeelemente

Formularelemente der Gruppe `<input>` benötigen das `type`-Attribut, um korrekt angezeigt zu werden. Unterstützt wird praktisch der gesamte Vorrat an HTML5-Typen:

- text
- password
- datetime
- datetime-local
- date
- month
- time
- week
- number
- email
- url
- search
- tel
- color

Input-Elemente

```
<input type="text"
       class="form-control"
       placeholder="Text input">
```

Textbereiche – also mehrzeilige Eingabefelder – nutzen dagegen folgenden Code:

```
<textarea class="form-control" rows="3" cols="55"></textarea>
```

Kontrollkästchen (Checkboxes) und Auswahlfelder (Radio Buttons) funktionieren wie bei Standard-HTML. Das Sperren mit dem Attribut `disabled` wird unterstützt. Damit das Sperren nicht nur für das Element selbst funktioniert, sondern auch das assoziierte Label betrifft, kann die Klasse `.disabled` benutzt werden. Diese Klasse ist auch im Zusammenhang mit `.radio`, `.radio-inline`, `.checkbox`, `.checkbox-inline` oder `<fieldset>` einsetzbar, die alle dazu dienen, den umschließenden Container passend zu formatieren.

Beispiel 4-12: Kontrollkästchen und Optionsfelder (Form_Elements.html)

```
<form class="row">
  <div class="checkbox">
    <label>
      <input type="checkbox" value="">
      Option Eins
    </label>
  </div>
  <div class="checkbox disabled">
    <label>
      <input type="checkbox" value="" disabled>
      Option Zwei (deaktiviert)
    </label>
  </div>

  <div class="radio">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios1" value="option1" checked>
      Option Eins
    </label>
  </div>
  <div class="radio">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios2" value="option2">
      Option Zwei
    </label>
  </div>
  <div class="radio disabled">
    <label>
      <input type="radio" name="optionsRadios"
             id="optionsRadios3" value="option3" disabled>
      Option Drei
    </label>
  </div>
</form>
```

Das `<label>`-Element umschließt hier das Optionsfeld bzw. das Kontrollkästchen. Auf diese Weise löst ein Klick auf die Beschriftung bzw. das Label die Aktion aus, was eine angenehmere Benutzererfahrung bietet.

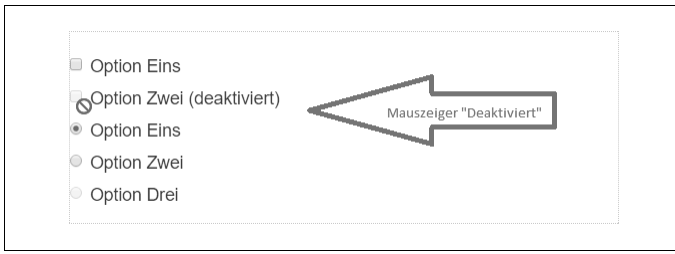

Abbildung 4-13: Kontrollkästchen und Optionsfelder

Die Klassen `.checkbox-inline` und `.radio-inline` ermöglichen eine Aufreihung von Elementen nebeneinander.

Beispiel 4-13: Kontrollkästchen und Optionsfelder nebeneinander (Form_ElementsHor.html)

```
<form class="row">
  <label class="checkbox-inline">
    <input type="checkbox" id="inlineCheckbox1" value="option1"> 1
  </label>
  <label class="checkbox-inline">
    <input type="checkbox" id="inlineCheckbox2" value="option2"> 2
  </label>
  <label class="checkbox-inline">
    <input type="checkbox" id="inlineCheckbox3" value="option3"> 3
  </label>

  <label class="radio-inline">
    <input type="radio" name="inlineRadioOptions"
           id="inlineRadio1" value="option1"> 1
  </label>
  <label class="radio-inline">
    <input type="radio" name="inlineRadioOptions"
           id="inlineRadio2" value="option2"> 2
  </label>
  <label class="radio-inline">
    <input type="radio" name="inlineRadioOptions"
           id="inlineRadio3" value="option3"> 3
  </label>
</form>
```

Abbildung 4-14: Kontrollkästchen und Optionsfelder nebeneinander

Wenn das Standardverhalten benötigt wird, aber kein Text für das Label angezeigt werden soll, entfällt der Text, und das Label umschließt das Eingabefeld:

```
<div class="checkbox">
  <label>
    <input type="checkbox" id="blankCheckbox" value="1"
           aria-label="...">
  </label>
</div>
<div class="radio">
  <label>
    <input type="radio" name="blankRadio" id="blankRadio1"
           value="1" aria-label="...">
  </label>
</div>
```

Listbox- und Drop-down-Elemente

Auch Listen und Drop-downs (Aufklappmenüs) lassen sich wie gewohnt einsetzen. Allerdings wenden einige Browser interne Stile an, die nicht mit CSS beeinflusst werden können. Möglicherweise sind Menüs die bessere Wahl, um volle Kontrolle über das Design zu erhalten.

Einfache Aufklappmenüs erlauben die Auswahl einer der Optionen:

```
<select class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

Mit dem Attribut `multiple` ist es möglich, mehrere Optionen auszuwählen:

```
<select multiple class="form-control">
  <option>1</option>
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
```

Statische Texte im Formular

Hilfetexte und statische Bausteine werden mit `.form-control-static` in einem Abschnitt `<p>` definiert.

Beispiel 4-14: Statische Elemente im Layout (Form_Help.html)

```
<form class="row">
  <div class="form-group">
    <label class="col-sm-2 form-control-label">
      E-Mail
    </label>
    <div class="col-sm-10">
      <p class="form-control-static">email@example.com</p>
    </div>
  </div>
  <div class="form-group">
    <label for="inputPassword"
           class="col-sm-2 form-control-label">
      Kennwort
    </label>
    <div class="col-sm-10">
      <input type="password" class="form-control"
             id="inputPassword" placeholder="Password">
    </div>
  </div>
</form>
```

Abbildung 4-15: Statische Elemente im vertikalen Layout

Beispiel 4-15: Anzeigefelder (Form_HelpInline.html)

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only">E-Mail</label>
    <p class="form-control-static">email@example.de</p>
  </div>
  <div class="form-group">
    <label for="inputPassword2" class="sr-only">Kennwort</label>
    <input type="password" class="form-control"
           id="inputPassword2" placeholder="Kennwort">
  </div>
  <button type="submit" class="btn btn-secondary">
     Bestätigen
  </button>
</form>
```

Abbildung 4-16: Statische Elemente im horizontalen Layout

Verhalten der Formularelemente

Formularelemente reagieren dynamisch auf den Fokus. Dabei wird der Rahmen entfernt, und beim Erhalten des Fokus wird ein weicher Schatten eingeblendet. Auslöser ist die Pseudoklasse `:focus`.

Deaktivierte Steuerelemente benutzen `disabled` und werden etwas heller dargestellt:

```
<input class="form-control"
       id="disabledInput"
       type="text"
       placeholder="Gesperrter Inhalt..."
       disabled>
```

Mit `<fieldset>` und `disabled` lassen sich mehrere Elemente zusammen abschalten. Allerdings betrifft die Blockierung von Benutzeraktionen nur reguläre Formularelemente. Hyperlinks werden nicht abgeschaltet. Auch wenn Hyperlinks als Schaltflächen formatiert werden (`<a ... class="btn btn-*">`), verhalten sie sich weiter wie Links. Hier muss mit zusätzlichem JavaScript nachgeholfen werden.

Bootstrap sorgt allerdings für die korrekte Darstellung, das heißt, dass die Schaltfläche »grau« erscheint. Sie wird dennoch ohne weitere Maßnahmen reagieren.

Beispiel 4-16: Gesperrte Felder (Form_Disabled.html)

```
<form>
  <fieldset disabled>
    <div class="form-group">
      <label for="disabledTextInput">Gesperrtes Feld</label>
      <input type="text" id="disabledTextInput"
             class="form-control"
             placeholder="Gesperrtes Feld">
    </div>
    <div class="form-group">
      <label for="disabledSelect">Gesperrtes Menü</label>
      <select id="disabledSelect" class="form-control">
        <option>Option A gesperrt</option>
        <option>Option B gesperrt</option>
      </select>
    </div>
    <div class="checkbox">
      <label>
        <input type="checkbox"> Kontrollkästchen
      </label>
    </div>
    <button type="submit" class="btn btn-primary">Senden</button>
  </fieldset>
</form>
```

Abbildung 4-17: Gesperrte Elemente

Neben dem Abschalten von Steuerelementen lässt sich auch der sehr ähnliche Zustand »Nur lesen« (readonly) einstellen. Er basiert auf dem HTML-Attribut `readonly`. Das Element wird heller dargestellt und sieht damit erst mal wie bei `disabled` aus, allerdings bleibt der Mauszeiger unverändert und zeigt nicht das »gesperrt-Icon« für den »disabled«-Cursor.

Beispiel 4-17: Nur-lesen-Modus (Ausschnitt aus Form_ReadOnly.html)

```
<input class="form-control"
       type="text"
       placeholder="Disabled Field"
       readonly>
```

Insbesondere das Verhalten der Aufklapplisten ist anders – im Gegensatz zu `disabled` lassen sich die Elemente anwählen und damit ansehen. Eine Auswahl beim Senden des Formulars gelingt dennoch nicht.

Abbildung 4-18: Elemente im Nur-lesen-Modus

Validierungsinformationen in Formularen

Bootstrap nutzt für Zustände die in HTML5 definierten Pseudoklassen `:invalid` und `:valid`. Damit beim ersten Anzeigen des Formulars nicht gleich alle Fehler erscheinen, muss dem `<form>`-Element die Klasse `.was-validated` hinzugefügt werden, was üblicherweise durch Servercode erfolgt. Die Initiierung geschieht dagegen durch `.needs-validation`.

Elemente, die die Klasse `.form-control-label` und `.form-control` enthalten, werden passend zum Status modifiziert.

Barrierefreiheit

Die Farbgestaltung der Statusangaben kann von Menschen mit Farbenblindheit möglicherweise nicht unterschieden werden. Deshalb sollte immer eine zusätzliche Angabe gemacht werden. Dies erfolgt durch Hilfetexte, Symbole und `.sr-only`-Bereiche. Elemente, die sich nicht anpassen lassen, sollten `aria-invalid="true"` tragen.

Beispiel 4-18: Semantische Informationen in Eingabefeldern (Form_Semantic.html)

```
<form class="needs-validation" novalidate>
  <div class="form-group">
    <label for="input1">
      Erfolgsmeldung
    </label>
    <input type="text" class="form-control"
           id="input1" required />
    <div class="valid-feedback text-success">Okay</div>
  </div>
  <div class="form-group">
    <label for="input2">
      Eine Warnung
    </label>
    <input type="text" value="Test" class="form-control"
           id="input2" required />
    <div class="invalid-feedback text-warning">Nicht Okay</div>
  </div>
  <div class="form-group">
    <label for="input3" >
      Fehlerbedingung
    </label>
    <input type="text" class="form-control"
           id="input3" required />
  </div>
  <div class="custom-checkbox">
    <label class="col-form-label">
      <input type="checkbox" id="checkboxSuccess"
             value="option1" />
```

```
      Erfolgsmeldung
    </label>
  </div>
  <div class="checkbox">
    <label class="col-form-label">
      <input type="checkbox" required
             id="checkboxWarning" value="option1" />
      Warnung
    </label>
  </div>
  <div class="checkbox">
    <label class="col-form-label">
      <input type="checkbox" id="checkboxError"
             value="option1" required />
      Fehlerbedingung
    </label>
  </div>
  <button class="btn btn-primary" type="submit">
    Testformular
  </button>
</form>
```

Die Steuerung erfordert JavaScript, wenn keine serverseitige Unterstützung vorliegt. Im Beispiel wird dieser Code genutzt:

```
<script>
  (function() {
    'use strict';
    window.addEventListener('load', function() {
      // Alle Formulare
      var forms = document.getElementsByClassName(
        'needs-validation');
      // Submit erkennen
      var validation = Array.prototype.filter.call(forms,
        function(form) {
          form.addEventListener('submit', function(event) {
            if (form.checkValidity() === false) {
              event.preventDefault();
              event.stopPropagation();
            }
          form.classList.add('was-validated');
        }, false);
      });
    }, false);
  })();
  </script>
```

Dies ist reines jQuery. Wenn Sie ein anderes Framework wie Angular oder React verwenden, muss der Code angepasst werden!

Abbildung 4-19: Semantische Formularelemente

Optional lassen sich die Zustände mit Symbolen aufwerten. Dies geschieht durch ein Symbol aus dem Paket *FontAwesome* am rechten Ende der Meldungsbereiche. Diese Symbole funktionieren nur, wenn `<input class="form-control">` eingesetzt wird.

Das folgende Beispiel zeigt, wie ein vollständig barrierefreies Formular aussehen kann:

Beispiel 4-19: Semantisch und barrierefrei (Form_AriaSemantic1.html)

```
<form>
  <div class="form-group text-success has-feedback">
    <label class="form-control-label text-success"
           for="inputSuccess2">Erfolg</label>
    <input type="text"
           class="form-control form-control-success"
           id="inputSuccess2"
           aria-describedby="inputSuccess2Status">
    <span id="inputSuccess2Status" class="sr-only">
      (Erfolg)</span>
```

```
    </div>
    <div class="form-group text-warning has-feedback">
      <label class="form-control-label text-warning"
             for="inputWarning2">Mit Warnung</label>
      <input type="text" class="form-control form-control-warning"
             id="inputWarning2"
             aria-describedby="inputWarning2Status">
      <span id="inputWarning2Status" class="sr-only">
      (Warnung)</span>
    </div>
    <div class="form-group text-danger has-feedback">
      <label class="form-control-label text-danger"
             for="inputError2">Mit Fehler</label>
      <input type="text" class="form-control form-control-danger"
             id="inputError2"
             aria-describedby="inputError2Status">
      <span id="inputError2Status" class="sr-only">
      (Fehler)</span>
    </div>
    <div class="form-group text-success has-feedback">
      <label class="form-control-label"
             for="inputGroupSuccess1">Erfolgreiche Gruppe</label>
      <div class="input-group">
        <span class="input-group-addon">@</span>
        <input type="text" class="form-control form-control-success"
               id="inputGroupSuccess1"
               aria-describedby="inputGroupSuccess1Status">
      </div>
      <span id="inputGroupSuccess1Status" class="sr-only">
      (Erfolg)</span>
    </div>
  </form>
```

Gegenüber Bootstrap 3 gibt es hier einige Änderungen. Zum einen wurden die Klassennamen vereinheitlicht. Die Klasse `.control-label` heißt jetzt `.form-control-label`. Sie regelt nur noch den Abstand, nicht mehr die Farbe der Bezeichnung. Wenn die Label auch farblich hervorgehoben werden sollen, muss mit `.text-success` oder `.text-danger` usw. gearbeitet werden.

Die Glyphicons entfallen komplett, und Symbole werden allein durch die semantischen Klassen `.form-control-success`, `.form-control-error` usw. erzeugt.

Abbildung 4-20: Semantische Elemente mit ARIA-Unterstützung

Symbole in Formularen, die einzeilig oder horizontal ausgelegt sind, werden folgendermaßen definiert:

Beispiel 4-20: Semantisch und barrierefrei (Form_AriaSemantic.html)

```
<form>
  <div class="form-group text-success has-feedback">
    <label class="form-control-label col-sm-3"
           for="inputSuccess3">Erfolg</label>
    <input type="text"
           class="form-control form-control-success"
           id="inputSuccess3"
           aria-describedby="inputSuccess3Status">
    <span id="inputSuccess3Status" class="sr-only">(Erfolg)
    </span>
  </div>
  <div class="form-group text-danger has-feedback">
    <label class="form-control-label col-sm-3"
           for="inputGroupError2">Fehler</label>
    <div class="input-group">
      <div class="input-group-prepend">
          <span class="input-group-text">@</span>
      </div>
      <input type="text"
             class="form-control form-control-danger"
             id="inputGroupError2"
             aria-describedby="inputGroupSuccess2Status">
```

```
    </div>
    <span id="inputGroupSuccess2Status" class="sr-only">(Fehler)
    </span>
  </div>
</form>
```

Abbildung 4-21: Semantische Elemente mit ARIA-Unterstützung

Hier ein Formular mit einer Erfolgsmeldung:

Beispiel 4-21: Semantische Meldungen (Form_Success.html)

```
<form class="form-inline">
  <div class="form-group has-feedback">
    <label class="form-control-label" for="inputSuccess4">
       Erfolg
    </label>
    <input type="text" class="form-control"
           id="inputSuccess4"
           aria-describedby="inputSuccess4Status">
    <div class="input-group-append border-left-0 bg-white border">
      <span class="fa fa-check form-control-feedback p-2"
            aria-hidden="true"></span>
    </div>
    <span id="inputSuccess4Status" class="sr-only">(Erfolg)</span>
  </div>
</form>
<form class="form-inline">
  <div class="form-group has-feedback">
    <label class="form-control-label" for="inputGroupSuccess3">
      Erfolgreiche Gruppe
    </label>
    <div class="input-group">
      <div class="input-group-prepend">
        <span class="input-group-text">@</span>
```

```
      </div>
      <input type="text" class="form-control border-right-0"
             id="inputGroupSuccess3"
             aria-describedby="inputGroupSuccess3Status">
      <div class="input-group-append border-left-0 bg-white border">
      <span class="fa fa-close form-control-feedback p-2"
            aria-hidden="true"></span>
    </div>
  </div>
  <span id="inputGroupSuccess3Status" class="sr-only">
     (Erfolg)
  </span>
</form>
```

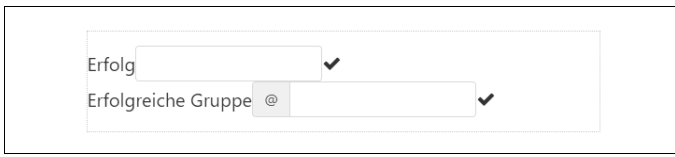

Abbildung 4-22: Semantische Meldungen

Optionale Symbole mit versteckten `.sr-only`-Abschnitten unterstützen Screenreader. Das Symbol wird automatisch passend ausgerichtet:

Beispiel 4-22: Meldungen für Screenreader (Form_SuccessSR.html)

```
<form class="form-inline">
  <div class="form-group text-success has-feedback">
    <label class="form-control-label sr-only"
           for="inputSuccess5">Hidden label</label>
    <input type="text" class="form-control"
           id="inputSuccess5"
           aria-describedby="inputSuccess5Status">
    <span class="fa fa-check form-control-feedback"
          aria-hidden="true"></span>
    <span id="inputSuccess5Status" class="sr-only">
    (Erfolg)</span>
  </div>
  <div class="form-group text-success has-feedback">
    <label class="form-control-label sr-only"
           for="inputGroupSuccess4">Erfolgreiche Gruppe</label>
    <div class="input-group">
      <div class="input-group-prepend">
```

```
            <span class="input-group-text">@</span>
        </div>
        <input type="text"
               class="form-control"
               id="inputGroupSuccess4"
               aria-describedby="inputGroupSuccess4Status">
      </div>
      <span class="fa fa-check form-control-feedback"
            aria-hidden="true"></span>
      <span id="inputGroupSuccess4Status" class="sr-only">
      (Erfolg)</span>
    </div>
  </form>
```

Formularelemente im Raster

Die Formatierung von Breite und Höhe wird mithilfe der üblichen Techniken vorgenommen. Für die Breite wird auf das Raster zurückgegriffen mit Klassen wie `.col-lg-3` oder `.col-md-5`.

Beispiel 4-23: Anordnung im Raster (Form_Grid.html)

```
<form>
  <div class="row">
    <div class="col-sm-2">
      <input type="text" class="form-control"
             placeholder="2 columns">
    </div>
    <div class="col-sm-3">
      <input type="text" class="form-control"
             placeholder="3 columns">
    </div>
    <div class="col-sm-4">
      <input type="text" class="form-control"
             placeholder="4 columns">
    </div>
  </div>
</form>
```

Abbildung 4-23: Gesperrte Elemente

Anpassung der Feldhöhe

Die Höhe nutzt eigene Klassen, die eine passende Ausdehnung ergeben: `.input-<xx></xx>`. Die Gruppe muss ebenfalls angepasst werden, was mittels der Klasse `.input-group-<xx></xx>` passiert. Für den Platzhalter `<xx>` setzen Sie `sm`, `md` oder `lg` ein. `xs` ist hier der Standard, und dafür gibt es keine extra Klasse.

Beispiel 4-24: Größe der Elemente (Form_ElementsSize.html)

```
<form>
  <div class="input-group-lg">
    <input class="form-control input-lg"
           type="text" placeholder="Groß">
  </div>
  <div class="input-group-md">
    <input class="form-control"
           type="text" placeholder="Normal">
  </div>
  <div class="input-group-sm">
    <input class="form-control input-sm"
           type="text" placeholder="Klein">
  </div>
  <div class="input-group-lg">
    <select class="form-control input-lg">
      <option>1</option>
      <option>2</option>
      <option>3</option>
    </select>
  </div>
  <div class="input-group-md">
    <select class="form-control">
      <option>1</option>
      <option>2</option>
      <option>3</option>
    </select>
  </div>
  <div class="input-group-sm">
    <select class="form-control input-sm">
      <option>1</option>
      <option>2</option>
      <option>3</option>
    </select>
  </div>
</form>
```

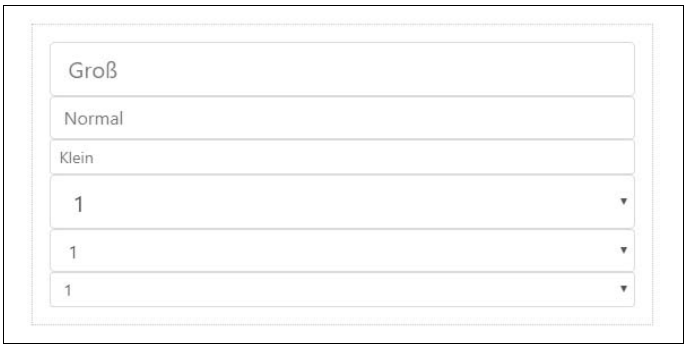

Abbildung 4-24: Elemente in verschiedenen Größen

Die Größe horizontal laufender Formulare lässt sich zentral kontrollieren. Dazu wird die Klasse `.form-group-lg` oder `.form-group-sm` genutzt.

Beispiel 4-25: Größe der Elemente (Form_ElementsSizeHor.html)

```
<form>
  <div class="form-group row">
    <label class="col-sm-2 form-control-label"
           for="formGroupInputLarge">Großes Label</label>
    <div class="col-sm-10">
      <input class="form-control form-control-lg" type="text"
             id="formGroupInputLarge" placeholder="Große Eingabe">
    </div>
  </div>
  <div class="form-group row">
    <label class="col-sm-2 form-control-label small"
           for="formGroupInputSmall">Kleines Label</label>
    <div class="col-sm-10">
      <input class="form-control form-control-sm" type="text"
             id="formGroupInputSmall" placeholder="Kleine Eingabe">
    </div>
  </div>
</form>
```

Abbildung 4-25: Elemente ausgerichtet

Hilfetexte in Formularen

Hilfetexte für Formulare werden immer wieder benötigt. Zum einen werden sie natürlich angezeigt, zum anderen unterstützen sie für barrierefreie Seiten auch Screenreader. Damit das funktioniert, sollten die passenden `aria`-Attribute immer mit den Bootstrap-Klassen zusammen eingesetzt werden.

Beispiel 4-26: Hilfetexte in Formularen (Form_ElementsHelp.html)

```
<form>
  <div class="form-group form-group-lg">
    <label for="inputHelpBlock">Ihre Eingabe</label>
    <input type="text" id="inputHelpBlock" class="form-control"
           aria-describedby="helpBlock">
    <span id="helpBlock" class="text-gray-dark small">
      Sie haben etwas eingegeben.</span>
  </div>
</form>
```

Die in Bootstrap 3 vorhandene Klasse `.help-block` gibt es nicht mehr. Sie erzielen diesen Effekt mit `.text-gray-dark` und `.small`, haben aber mit weiteren Farben und Größen nun viel mehr Optionen, ohne weitere Klassen einsetzen zu müssen.

Abbildung 4-26: Hilfetexte

Schaltflächen

Schaltflächen (Buttons) sind in der einen oder anderen Form in jeder Webanwendung zu finden. Bootstrap sieht in Schaltflächen ein Gestaltungselement, das von technischen Rahmenbedingungen losgelöst ist. Technisch kann eine Schaltfläche nur ein Formular absenden, was praktisch immer eine POST-Anfrage auslöst. Wird dagegen ein Zugriff auf den Server via GET benötigt, kommt ein Hyperlink zum Einsatz. Gestalterisch stellt Bootstrap nun beides auf eine Stufe. Schaltflächen lassen sich mit `<a>`, `<button>` oder `<input>` erstellen.

Beispiel 4-27: Schaltflächen (Form_Buttons.html)

```
<a class="btn btn-secondary" href="#" role="button">
  Link
</a>
<button class="btn btn-secondary" type="submit">
   Schaltfläche
</button>
<input class="btn btn-secondary"
       type="button" value="Eingabe">
<input class="btn btn-secondary"
       type="submit" value="Absenden">
```

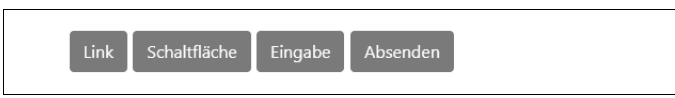

Abbildung 4-27: Schaltflächen – mehrere technische Varianten mit derselben Gestaltung

Es gibt jedoch die Einschränkung, dass in Navigationsleisten, die mit `.nav` oder `.navbar` aufgebaut sind, nur das neutrale Element `<button>` zulässig ist.

Werden `<a>`-Tags als Schaltfläche benutzt, sollten diese lediglich der Seitennavigation dienen und nicht ihre ursprüngliche Funktion als Hyperlink auslösen, weil das nicht das vom Benutzer erwartete Verhalten ist. Darüber hinaus sollte die Funktion durch `role="button"` unterstrichen werden. Auch dann ist es fraglich, ob das eine gute Idee ist, weil nicht bei allen Browsern sichergestellt werden kann, dass sich deren Funktion wie erwartet verhält. Nutzen Sie im Zwei-

felsfall für Schaltflächen – wann immer dies möglich ist – bevorzugt `<button>`.

Semantische Schaltflächen

Schaltflächen gibt es in sechs Varianten mit semantischer Ausprägung:

- *Secondary*: Standard, grau, allgemeine Funktion.
- *Primary*: Löst die Hauptfunktion aus, visueller Effekt durch blaue Farbe verstärkt.
- *Success*: Erfolg, grün, positive oder bejahende Aktion.
- *Info*: Information, violett, verdeutlichende Hervorhebung kritischer oder spezieller Aktionen.
- *Warning*: Warnung, orange, kritische oder komplexe Aktion, bei der Vorsicht geboten ist.
- *Danger*: Gefahr, rot, gefährliche oder unumkehrbare Aktion, negativ.

Beispiel 4-28: Semantische Schaltflächen

```
<button type="button" class="btn btn-secondary">Standard</button>

<button type="button" class="btn btn-primary">Primär</button>

<button type="button" class="btn btn-success">Erfolg</button>

<button type="button" class="btn btn-info">Information</button>

<button type="button" class="btn btn-warning">Warnung</button>

<button type="button" class="btn btn-danger">Gefahr</button>
```

Abbildung 4-28: Semantische Schaltflächen

Wie bereits zuvor ist die Zuordnung einer ansatzweise semantischen Bedeutung mittels Farben auf Umgebungen beschränkt, die

Farben uneingeschränkt wiedergeben können. Zusätzliche Label, die mit `.sr-only` versehen sind, sollten die beabsichtigte Wirkung unterstreichen.

Darüber hinaus gibt es drei Varianten mit nicht semantischer Ausrichtung, die vor allem der Gestaltung dienen:

- *Light*: Helle Darstellung, schwarz auf weiß.
- *Dark*: Dunkle Darstellung, weiß auf schwarz.
- *Link*: Darstellung als Hyperlink (Verhalten bleibt eine Schaltfläche).

Größe und Aussehen

Für größere oder kleinere Schaltflächen sollten die Klassen `.btn-lg` bzw. `.btn-sm` benutzt werden. Sie ergänzen die Basisklasse `.btn` und sind unabhängig von der Farbgebung. Die Standardgröße eignet sich bereits für die Touchbedienung. Mausgesteuerte Seiten werden mit kleineren Schaltflächen besser aussehen. Die Größen `xs` und `xl` gibt es nicht, `md` ist der Standard, und dafür gibt es auch keine explizite Definition.

```
<p>
  <button type="button" class="btn btn-primary btn-lg">
     Groß
  </button>
  <button type="button" class="btn btn-secondary btn-lg">
     Groß
  </button>
</p>
<p>
  <button type="button" class="btn btn-primary">
     Primär
  </button>
  <button type="button" class="btn btn-secondary">
     Standard
  </button>
</p>
<p>
  <button type="button" class="btn btn-primary btn-sm">
     Klein
  </button>
```

```
  <button type="button" class="btn btn-secondary btn-sm">
     Klein
  </button>
</p>
```

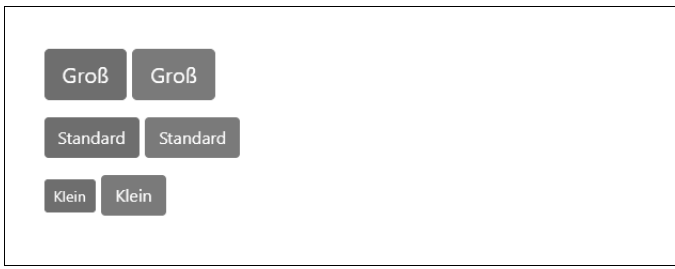

Abbildung 4-29: Größe der Schaltflächen

Manchmal sollen Schaltflächen das umgebende HTML-Element, in dem sie sind, komplett ausfüllen. Dazu dient die Klasse `.btn-block`.

Beispiel 4-29: Schaltflächen horizontal erweitert (Form_ButtonsContainer.html)

```
<button type="button"
        class="btn btn-primary btn-lg btn-block">
  Block
</button>
<button type="button"
        class="btn btn-secondary btn-lg btn-block">
  Block
</button>
```

Beachten Sie hier, dass das nur auf die Breite wirkt. Eine passende Höhe muss separat mit `.btn-lg` usw. erreicht werden.

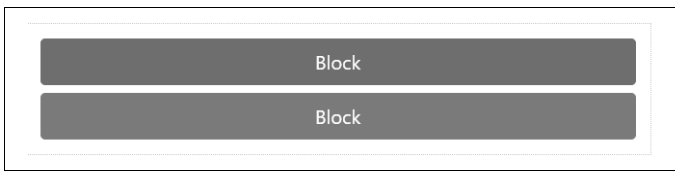

Abbildung 4-30: Schaltflächen horizontal erweitert

Neben der Basisfarbe können Schaltflächen Zustände haben. Der Zustand »gedrückt« wird mit `:active` bzw. bei formatierten Hyperlinks auch mit `.active` erreicht. Fügen Sie für eine barrierefreie Ausprägung `aria-pressed="true"` hinzu.

```
<button type="button" class="btn btn-primary btn-lg active">
  Primär
</button>
<button type="button" class="btn btn-secondary btn-lg active">
  Standard
</button>
```

Bootstrap 4 bietet auch die Möglichkeit, umrandete Schaltflächen darzustellen. Damit wird die massive optische Wirkung der klassischen Schaltflächen vermieden. Bisher profitierten die Elemente von einem schwachen Farbverlauf, der einen leichten 3-D-Effekt erzeugte. Die in neueren Websites oft benutzten flachen, reduzierten Gestaltungen wirken damit aber überfrachtet. Hier greifen die »Outline«-Schaltflächen:

Beispiel 4-30: Semantische Schaltflächen mit Rahmen

```
<button type="button"
        class="btn btn-outline-secondary">Standard</button>

<button type="button"
        class="btn btn-outline-primary">Primär</button>

<button type="button"
        class="btn btn-outline-success">Erfolg</button>

<button type="button"
        class="btn btn-outline-info">Information</button>

<button type="button"
        class="btn btn-outline-warning">Warnung</button>

<button type="button"
        class="btn btn-outline-danger">Gefahr</button>
```

Abbildung 4-31: Semantische Schaltflächen mit Rahmen

Zustände

```
<a href="#"
   class="btn btn-primary btn-lg active"
   role="button">
  Primär
</a>
<a href="#"
   class="btn btn-secondary btn-lg active"
   role="button">
  Link
</a>
```

Der Zustand »deaktiviert« (disabled) zeigt an, dass die Schaltfläche nicht gedrückt werden kann. Dazu wird das HTML-Attribut `disabled` benutzt.

```
<button type="button"
        class="btn btn-lg btn-primary"
        disabled="disabled">Primär</button>
<button type="button"
        class="btn btn-secondary btn-lg"
        disabled="disabled">Standard</button>
```

Für als Hyperlinks formatierte Schaltflächen kann die Klasse `.disabled` benutzt werden.

```
<a href="#"
   class="btn btn-primary btn-lg disabled"
   role="button">Primär</a>
<a href="#"
   class="btn btn-secondary btn-lg disabled"
   role="button">Link</a>
```

Abbildung 4-32: Diverse Zustände einer Schaltfläche

Nur CSS

Beachten Sie, dass die Abschaltung von Elementen mit CSS-Klassen lediglich Kosmetik ist. Auch die Nutzung von `pointer-events: none` verhindert nicht, dass sich solche Elemente mit der Tastatur anwählen und auslösen lassen. Sie müssen hier fast immer mit eigenem JavaScript dafür sorgen, dass die Funktion mit der Darstellung übereinstimmt.