online Übungssystem & online Prüfungssystem

Formular-Eingabevalidierung im Online-Übungssystem

Autor:
Immo Schulz-Gerlach, ZDI

Version:
2.1 – 12. August 2024

FernUniversität in Hagen

Inhaltsverzeichnis

Überblick 3
- Kurzreferenz Formularklassen 3
- Kombination beider Validierungen 4

1. Formular-Eingabevalidierung per HTML 5 5
- Formatierung der Eingabefelder durchs Online-Übungssystem 5
- Beispiele für Umsetzung von Validierungen mit HTML 5 6
  - Pflichtfelder 6
  - Zahlenfelder 7
    - Inputmode 8
    - Typische Beispiele für Zahlenfelder 10
- Checkboxen als Pflichtfelder 12

2. Formular-Eingabevalidierung per Java-Script (deprecated) 14
- Einrichtung 14
  - Hinweise 15

Überblick

Im Jahr 2011 wurde erstmals eine clientseitige Eingabevalidierung im Online-Übungssystem ermöglicht. Dafür zuständig war eine JavaScript-Library, die eben vor einem Submitversuch die entsprechend gekennzeichneten Eingabefelder auf Gültigkeit der Eingaben überprüfte, also z.B. leere Pflichtfelder meldete oder Texteingaben in Zahlenfeldern.

Weil die Ansteuerung dieser JavaScript-Validierung (allein durch Einträge im `class`-Attribut von HTML-Tags) eine Übungssystem-spezifische Eigenentwicklung war, wurde die Version 1 dieses Handbuchs dazu geschrieben.

Inzwischen wird kein JavaScript mehr für Formularvalidierung mehr benötigt, der HTML-5-Standard sieht eigene HTML-Attribute für die Deklaration von zu validierenden Constraints vor. Dass kein JavaScript mehr benötigt wird, ist nur *einer* der Vorteile der HTML-5-Syntax. Darüber hinaus bieten die neuen Feldtypen weitere Vorteile wie Eingabeunterstützungen. So kann ein Zahlenfeld z.B. in Desktop-Browsern (für Maussteuerung) vom Browser direkt mit „Spin-Buttons“, also kleinen Pfeil-Icons zum De- und Inkrementieren des Zahlenwerts versehen werden, und für Smartphone-Browser kann gezielt eine Zehnertastatur mit oder ohne Komma-Taste statt einer für Textfelder sonst üblichen Buchstabentastatur auf dem Touchscreen eingeblendet werden.

Außerdem ist es von Vorteil, dass (weitestgehend) keine Übungssystem-spezifische Syntax mehr benötigt wird, sondern eben direkt auf den systemübergreifend anwendbaren HTML-Standard gesetzt werden kann.

Daher wird heute eher der Einsatz der HTML-5-Validierung empfohlen, die auch mehr Möglichkeiten bietet als die ältere JavaScript-Validierung.

Das erste Kapitel dieses Handbuchs geht daher zunächst auf den Einsatz von HTML-5-Validierung ein. Da es sich um einen gut dokumentierten Standard handelt, wird dieses Übungssystem-Handbuch nicht alle Möglichkeiten ausführlich beschreiben, sondern eher ein paar simple Beispiele für die wichtigsten Szenarien, insbesondere für die zuvor schon mit der JavaScript-Variante möglichen Validierungen zusammenstellen und auf die Übungssystem-eigenen Mechanismen zur Aktivierung bestimmen Funktionen/Hervorhebungen eingehen.

Die Dokumentation zur bisherigen JavaScript-Validierung (die nach wie vor funktioniert), finden Sie anschließend im Kapitel 2.

Kurzreferenz Formularklassen

Vorab eine Übersicht der möglichen Klassen, die im `class`-Attribut eines `<form>`-Tags eingetragen werden können. Die folgenden Kapitel gehen näher darauf ein.

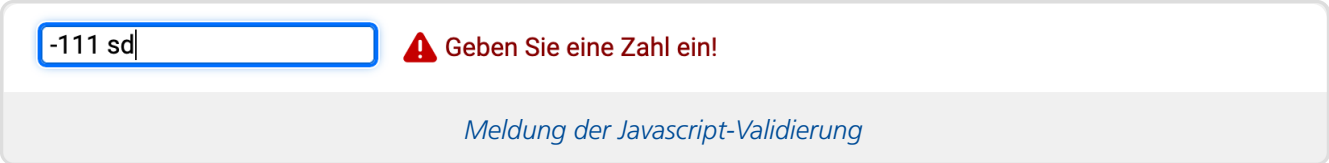
Klasse	Funktion
`validation-styles`	Aktiviert Übungssystem-eigene CSS-Stile und JavaScripte für den Einsatz mit HTML-5-basierter Eingabevalidierung (siehe Kapitel 1).
`required-checkbox-groups`	Aktiviert JavaScript zur Unterstützung von Checkbox-Gruppen als Pflichtfeld, siehe Checkboxen als Pflichtfelder. In der Regel mit `validation-styles` in Kombination zu verwenden.
`validated`	Aktiviert die bisherige JavaScript-Validierung, siehe Kapitel 2. Sollte *nicht* mit den beiden obigen Klassen kombiniert werden!

Das Online-Übungssystem wird Ihre Aufgabenseiten automatisch nach diesen Klassen durchsuchen und immer, wenn es so markiertes Formular findet, die benötigten JavaScript-Dateien automatisch einbinden (Script-Tags ins ausgelieferte HTML injizieren)[1].

Kombination beider Validierungen

Prinzipiell können Sie HTML-5-Tags mit der JavaScript-basierten Validierung kombinieren, wenn Ihnen die eingeschränkteren Möglichkeiten der letzten ausreichen und Sie deren Einblendung von Texten bevorzugen.

So blendet die JavaScript-Validierung ihre Fehlermeldung eigenständig, browserunabhängig und dauerhaft sichtbar hinter dem Eingabefeld ein:



Meldung der Javascript-Validierung

Bei der HTML-5-Variante dagegen sorgen die Browser selbst für eine Anzeige von Fehlerhinweisen, was je nach Browser durchaus unterschiedlich aussehen kann, z.B.:

Wenn Sie beides kombinieren möchten, tragen Sie im `form`-Tag nur die Klasse `validated` ein und verzichten Sie auf die anderen Klasse aus obiger Übersichtstabelle! Für die Inputs können Sie dann z.B. HTML-5-Input-Typen verwenden und ihnen im `class`-Attribut dann zusätzlich noch die von der JavaScript-Lösung benötigten Klassen zuordnen (siehe 2. Kapitel).

1. Formular-Eingabevalidierung per HTML 5

HTML-5-Validierung können Sie prinzipiell jederzeit benutzen, dazu wird keine spezielle Funktionalität des Online-Übungssystems benötigt.

Sie müssen dazu lediglich Standard-HTML-Attribute zu Ihrem Formular (`form`-Tag) oder den Formularelementen (wie `input`-Tags) hinzufügen.

Hier ein paar Einstiegslinks zur Dokumentation in MDN:

- Form(ular)-Element <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form>
- Input-Element <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>
- Number-Input-Element <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/number#validation>
- Validierung <https://developer.mozilla.org/en-US/docs/Web/HTML/Constraint_validation>
- Tutorial zu Validierung <https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation>

Um allerdings Übungssystem-eigene CSS-Stile oder die Übungssystem-eigene Lösung für „Pflicht-Checkbox-Gruppen“ zu aktivieren, sind dem Formular-Tag (`<form>`) Übungssystem-spezifische Klassen im `class`-Attribut zuzuweisen (siehe Tabelle Kurzreferenz Formularklassen). Bei Vorhandensein mindestens einer dieser Klassen lädt das Online-Übungssystem ein eigenes kleines JavaScript mit Zusatzfunktionen hinzu.

Formatierung der Eingabefelder durchs Online-Übungssystem

Das Standard-CSS des Online-Übungssystems wird bereits versuchen, Input-Felder, deren Eingaben nicht die Constraints erfüllen (wie leere Pflichtfelder, Zahlenfelder mit Text darin oder Zahlenfelder mit einer Zahl außerhalb des erlaubten Werftenbereiches etc.), hervorzuheben, derzeit durch roten Rahmen.

Dieses Standard-CSS allein hat aber ein paar Nachteile:

- Es gibt z.B. keine Icons zur Markierung von fehlerhaft ausgefüllten Feldern oder leeren Pflichtfeldern.
- Und wenn Sie z.B. ein Aufgabenformular mit noch leeren Pflichtfeldern ausliefern, würden diese Pflichtfelder sofort (und nur) durch einen roten Rahmen gekennzeichnet.

Wenn Sie dagegen dem `<form>`-Tag des HTML-Formulars im `class`-Attribut die Klasse `validation-styles` hinzufügen, aktiviert das Online-Übungssystem zusätzlich per JavaScript insbesondere derzeit folgende Funktionen (Änderungen vorbehalten):

- Leere Pflichtfelder werden durch ein nachgestelltes Sternchen gekennzeichnet. Sobald etwas eingegeben wird (und eine gültige Eingabe darstellt), wird das Sternchen ausgeblendet.
- Eine rote Umrandung der leeren Pflichtfelder oder mit ungültigen Eingaben ausgefüllten Eingabefelder erfolgt erst nach einer Texteingabe in diese Felder (oder einem Einsendeversuch).
- Nach einem wegen nicht korrekt ausgefülltem Formular nicht erfolgreichem Einsendeversuch werden nicht nur alle fehlerhaft ausgefüllten Felder rot umrandet, sondern dann auch ein rotes „Warndreieck“-Symbol dahinter eingeblendet.

Beispiel für ein Formular mit `validation-styles`-Klasse, einem Pflichtfeld und einem Zahlenfeld (für beliebige Fließkommazahlen ≥ 0):

```
<form class="validation-styles"
action="$WebAssignServer/$Veranstaltername/Einsendung/$KursNr/$VersionsNr/$Aufg
abenheftNr/$AufgabenNr" method="post">
    …
    <input type="text" name="FeldA1" required>
    …
    <input type="number" inputmode="decimal" min="0" step="any" name="FeldA2">
    …
</form>
```

Beispiele für Umsetzung von Validierungen mit HTML 5

Bei den folgenden Beispielen wird davon ausgegangen (also nicht mehr extra mit angegeben), dass diese in einem Formularelement mit Klasse `validation-styles` stehen:

```
<form class="validation-styles"
action="$WebAssignServer/$Veranstaltername/Einsendung/$KursNr/$VersionsNr/$Aufg
abenheftNr/$AufgabenNr" method="post">
    …
    …Beispiel-Elemente…
    …
</form>
```

Die HTML-Validierung wird zwar auch ohne diese Class funktionieren, aber die empfohlene CSS-Darstellung wird dadurch aktiviert, siehe Formatierung der Eingabefelder durchs Online-Übungssystem

Pflichtfelder

Um ein Feld als Pflichtfeld zu kennzeichnen, versehen Sie es einfach mit dem Attribut `required` (als eigenständiges Attribut – nicht, wie es bei der alten JS-Lösung der Fall war, als Klasse), z.B.:

```
<input type="text" name="FeldA1" required>
```

Das geht prinzipiell mit beliebigen Input-Feldern, also insbesondere

- *Text-Feldern* wie in obigem Beispiel,
- mit *Zahlenfeldern* (wie im Folgenden beschrieben)
- oder mit *Textareas*.
- Ein Besonderheit von *Radiobuttons* ist, dass hier *mehrere* Input-Elemente mit gleichem Namen existieren. Um die Eigenschaft als Pflichtfeld zu beschreiben, also auszudrücken, dass einer der Radiobuttons einer Gruppe (also mit gleichem `name`-Attribut) ausgewählt sein muss, genügt es bereits, (mindestens) *einem* dieser Radio-Inputs das `required`-Attribut zu geben. Mit Blick auf CSS-Hervorhebung von Required-Fields ist es aber sinnvoll, das für *alle* zu tun:

  ```
  <input type="radio" name="FeldA2" value="A" required>
  <input type="radio" name="FeldA2" value="B" required>
  <input type="radio" name="FeldA2" value="C" required>
  ```

- Einen echten *Sonderfall* stellen dagegen *Checkboxen* dar. Wenn man in HTML5 eine Checkbox als `required` auszeichnet, heißt das, dass genau diese Checkbox angekreuzt sein muss, es ist aber „out of the box“ nicht möglich, zu einer *Gruppe* von Checkboxen auszudrücken, dass mindestens eine davon markiert sein muss. Dafür hat das Online-Übungssystem eine eigene Lösung realisiert, die im Folgenden in Abschnitt Checkboxen als Pflichtfelder gesondert erläutert wird.

Zahlenfelder

Um ein Feld als Feld für eine Zahleneingabe zu markieren, können Sie den Input-Type von `text` auf `number` ändern. Das hat verschiedene Auswirkungen, z.B.:

- Für Number-Felder werden Desktopbrowser typischerweise Spinner-Buttons einblenden, je nach Browser dauerhaft oder nur, wenn das Feld den Fokus hat (siehe Abbildungen weiter unten).
- Der Browser weiß, dass dies nicht einfach ein Text nach gewissem Format sein soll, sondern eine Zahl. Wenn Sie **Fließkommazahlen** erlauben möchten, wird der Browser z.B. die Anzeige der Zahl an die Landeseinstellungen anpassen (Komma als Dezimaltrenner), aber den Zahlenwert dann ans Übungssystem immer in der „internationalen“ bzw. amerikanischen Schreibweise (mit Punkt als Dezimaltrenner) übermitteln!
- Mit weiteren Attributen wie `min` und `max` können die Zahlenbereiche eingeschränkt werden.
- Über das `step`-Attribut (Default ist 1) lässt sich insb. die Schrittweite für die Spinner-Buttons festlegen, es schränkt aber auch die möglichen gültigen Zahlenwerte ein, siehe nachfolgende Punkte.
    - Falls Sie `min` und `max` in Verbindung mit `step` verwenden, gelten nur die Zahleneingaben als zulässig, die sich durch Addieren eines Vielfachen von `step` zum Minimum errechnen. Ein

      ```
      <input type="number" min="0" step="5">
      ```

      wird z.B. nur durch 5 teilbare Zahlen (0, 5, 10, 15, 20, …) akzeptieren.
    - Falls Sie Fließkommazahlen erlauben wollen (siehe auch `inputmode="decimal"` unten), legt ein `step`-Wert auch die mögliche Anzahl an Nachkommastellen fest. Mit `step="0.01"` z.B. sind beliebige zwei Nachkommastellen möglich, aber nicht mehr als zwei. Mit `step="0.05"` sind ebenfalls nur zwei Nachkommastellen möglich, wobei die zweite nur 5 oder 0 sein kann (zumindest bei ganzzahligem oder nicht angegebenem Minimum). Mit `step="0.5"` wiederum könnten nur ganze und halbe Zahlen eingegeben werden, also entweder ganze Zahlen oder Zahlen, die auf `",5"` enden.
    - Mit `step="any"` dagegen werden beliebig viele Nachkommastellen erlaubt. Die Spinner-Buttons werden in der Regel (abhängig vom Browser) den Wert um jeweils 1 de- bzw. inkrementieren (ob dabei bereits eingegebene Nachkommastellen beibehalten werden, also nur der Wert vor dem Komma erhöht oder erniedrigt wird, oder ob sie abgeschnitten werden, also der Wert zur nächstgrößeren bzw. nächstkleineren Ganzen Zahl geändert wird, ist wieder eine Entscheidung des Browsers).

Im Übrigen gibt es noch diverse weitere HTML-Attribute für Text- oder numerische Inputs, z.B. `pattern`, mit dem ein Textmuster als regulärer Ausdruck festgelegt werden kann.

Inputmode

Zusätzlich zu den oben genannten Attributen wie `type="number"`, um ein Eingabefeld überhaupt auf Zahleneingaben festzulegen, sowie Constraint-Attributen wie `min`, `max`, `step`, die den Wertebereich der möglichen Zahlen einschränken, kann es sinnvoll sein, *zusätzlich* das Attribut `inputmode` anzugeben, um einen Texteingabemodus dazu festzulegen. Das hat in Browsern auf typischen PCs mit Bildschirm, Tastatur und Maus keine Auswirkungen, kann aber auf gewissen Touchscreen-Geräten wie insbesondere Smartphones, die eine virtuelle Tastatur einblenden, ein abweichendes Tastaturlayout einstellen. Ohne Angabe wird ein Smartphone unter Umständen eine Standard-Texttastatur auch für Zahlenfelder einblenden, auf der die Eingabe von Zahlen vergleichsweise weniger komfortabel ist als z.B. auf einer reinen Zifferntastatur.

Für Zahleneingaben gibt es im Wesentlichen zwei Eingabemodi:

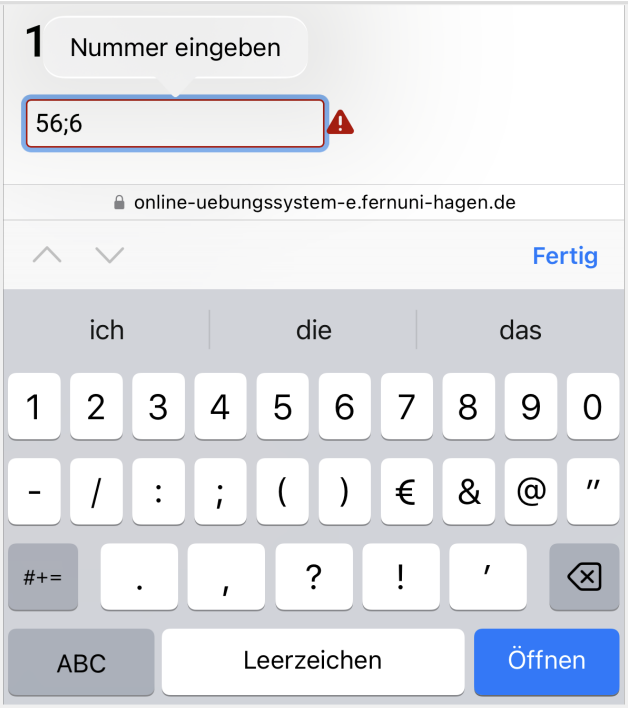
inputmode	Tastatur
`numeric`	Zifferntastatur. Je nach Gerät/Browser kann sie auch weitere Tasten enthalten, aber typisch sind ausschließlich Tasten für die Ziffern 0 bis 9. Damit lassen sich also im Zweifel keine Fließkommazahlen und auch keine Vorzeichen eingeben, also nur Ziffernfolgen / natürliche Zahlen.
`decimal`	Zifferntastatur, die ein zur im Betriebssystem eingestellten Region passendes Dezimaltrennzeichen eingeben kann, in Deutschland also ein Komma, in Amerika einen Punkt. Auch hier ist nicht sichergestellt, dass ein Vorzeichen eingegeben werden kann! Android-Tastaturen im Decimal-Modus umfassen m.W. eine Minus-Taste, bei iOS dagegen gibt es wirklich nur die Ziffern und das Komma, siehe folgende Abbildung.

Fazit:

- Für Eingabefelder, in die nur Ziffernfolgen/natürliche Zahlen eingegeben werden können sollen, empfiehlt es sich, auch eine Zifferntastatur einzublenden. Dann gibt es Sinn, auch für Nicht-Smartphone-Nutzer die 0 (oder eine größere/positive Zahl) als Minimum der möglichen Eingaben über das `min`-Attribut festzulegen: `type="number" min="0" inputmode="numeric"`
- Für Eingabefelder, in die zwar Fließkommazahlen eingegeben werden sollen, die aber ebenfalls nicht negativ werden dürfen, kann entsprechend `inputmode="decimal"` hinzugefügt werden.
  In iOS kann das z.B. so aussehen:

iOS-Bildschirmtastatur für Inputmode decimal (ohne Minus-Taste)

- Für alle Zahleneingabefelder, in denen negative Eingaben erlaubt sein sollen, sollte sicherheitshalber die Standard-Texttastatur verwendet werden. Auch wenn manche Geräte auch mit z.B. `decimal`-Inputmode die Eingabe des Minus-Zeichens ermöglichen, ist das doch nicht auf überall möglich, insb. – Stand Mai 2023 – nicht auf iPhones! Und das deckt sich auch mit dem HTML-Standard, der lediglich die Eingabemöglichkeiten für Ziffern und Dezimaltrenner fordert. (Auf solchen Geräten wie iPhones könnte man dann das Minus-Zeichen höchstens per Copy&Paste in die Eingabezeile kopieren, kann es aber nicht eintippen.)
  Zumindest auf iPhones gilt aber, dass bei Kombination von `type="number"` bei ganz fehlendem `inputmode`-Attribut zwar durchaus die Standardtastatur eingeblendet wird, darin aber immerhin der Zahlen- und Sonderzeichen-Eingabemodus vorausgewählt wird, während in Textfeldern (`type="text"`) oder bei expliziter Angabe von `inputmode="text"` diese Tastatur standardmäßig die Buchstaben-/Texteingabe vorselektieren wird.
  Der folgende Screenshot zeigt z.B. die Tastatur (Stand Mai 2023) in iOS für ein `<input type="number">` ohne Inputmode-Angabe:



Default-iOS-Bildschirmtastatur bei Zahlenfeldern

Das `inputmode`-Attirbut kann übrigens nicht nur mit `type="number"` kombiniert werden, sondern auch mit `type="text"`. Das kann z.B. sinnvoll sein, falls Sie Spinner-Buttons verhindern wollen oder möchten, dass das eingegebene nationale Dezimaltrennersymbol unverändert gesendet und nicht – wie es bei type=`number` der Fall wäre – vom Browser vor der Einsendung durch einen Punkt ersetzt wird.

Typische Beispiele für Zahlenfelder

1. Natürliche Zahlen / Ziffernfolgen (entspricht `class="digits"` bei der alten JavaScript-Validierung):

   ```
   <input type="number" inputmode="numeric" min="0" … >
   ```

   - Da kein `step`-Attribut angegeben wurde, gilt die Standard-Schrittweite von 1, was die Eingabemöglichkeit auf ganze Zahlen beschränkt.
   - `inputmode="numeric"` aktiviert die Zifferntastatur z.B. auf Smartphones (s.o.), die auf manchen Geräten keine Minustaste besitzt.
   - Das `min`-Attribut legt fest, dass negative Zahlen grundsätzlich nicht eingesendet werden dürfen (auch nicht, wenn die Minustaste vorhanden ist).
2. beliebige ganze Zahlen (entspricht `class="integer"` bei der alten JavaScript-Validierung):

   ```
   <input type="number" … >
   ```

   - Da kein `step`-Attribut angegeben wurde, gilt die Standard-Schrittweite von 1, was die Eingabemöglichkeit auf ganze Zahlen beschränkt.
   - Ohne `min`-Attribut können auch negative ganze Zahlen eingegeben werden.
   - Es wurde keine Zifferntastatur angefordert, damit die Eingabe des Minus-Vorzeichens auch auf allen Touchscreen-Geräten überhaupt möglich ist.
3. beliebige reelle Zahlen (entspricht `class="number"` bei der alten JavaScript-Validierung):

   ```
   <input type="number" step="any" … >
   ```

   - `step="any"` erlaubt beliebig viele Nachkommastellen.
   - Auch hier wird keine Zifferntastatur angefordert, da selbst die Dezimal-Tastatur (siehe nächstes Beispiel) nicht auf allen Endgeräten eine Minus-Vorzeicheneingabe ermöglicht!
4. nicht-negative reelle Zahlen:

   ```
   <input type="number" inputmode="decimal" min="0" step="any" … >
   ```

   - Gegenüber dem obigen Beispiel wurde hier wieder der Wertebereich auf nicht-negative Zahlen begrenzt, indem `min="0"` festgelegt wurde,
   - und nun kann dem Smartphone-Nutzer auch wieder gefahrlos entgegengekommen werden, indem die Standard-Bildschirmtastatur durch eine Zehnertastatur mit zusätzlicher Kommataste ersetzt wird.
5. Textfeld für natürliche Zahlen/Ziffernfolgen (verhindert die Einblendung von Spinner-Buttons):

```
<input type="text" inputmode="numeric" pattern="\d*">
```

- Hier wird dem Browser gesagt, dass er ein Textfeld anzeigen soll, in dem aber nur Text als gültig gilt, der ausschließlich aus Ziffern besteht. Letzteres wird über das `Pattern`-Attribut als regulärer Ausdruck festgelegt, wobei `\d` für eine Ziffer (digit) und `*` für „beliebig viele davon“ steht.

6. Textfeld für ganze Zahlen mit oder ohne Vorzeichen (wobei hier auch `+` als Vorzeichen explizit erlaubt wird):

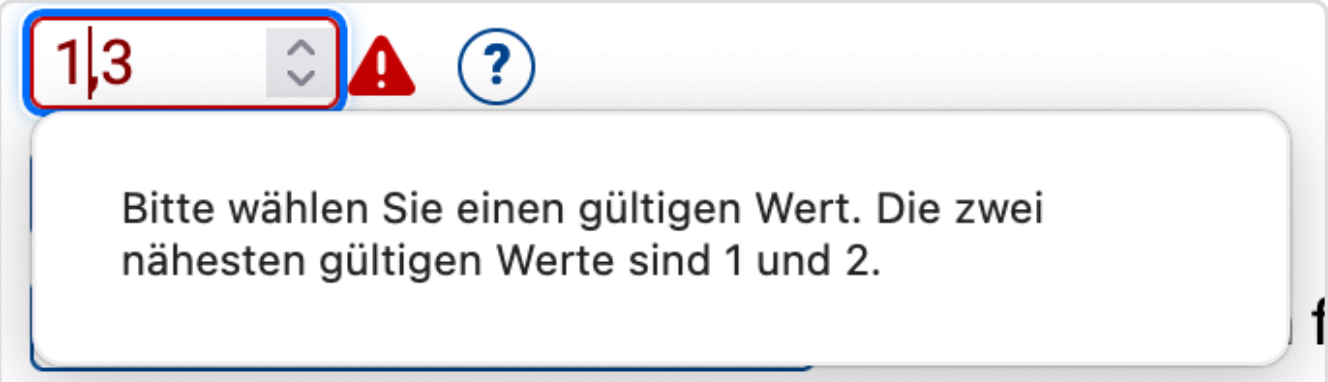
```
<input type="text" pattern="[+-]?\d+">
```

- Hier wird dem Browser gesagt, dass er ein Textfeld anzeigen soll, in dem aber nur Text als gültig gilt, der ausschließlich aus Ziffern und optional einem + oder - davor besteht. Ein Vorzeichen ohne Ziffern dahinter ist nicht gültig. (Das `\d+` steht für *mindestens eine* Ziffer; `[+-]` steht für "Plus oder Minus" und das Fragezeichen dahinter macht das Ganze optional, so dass das Vorzeichen auch weggelassen werden darf.)

Die beiden letzten Beispiele wurden der Vollständigkeit halber mit aufgeführt. Sie verhindern durch Verwenden eines Text- statt Zahlenfelders insbesondere das Einblenden von „Spinner-Buttons“ (also Pfeil-Buttons zum Erhöhen oder Verringern des angezeigten aktuellen Werts, siehe folgende Abbildung) in Desktop-Browsern.

Gerade bei Eingabefeldern für sehr große Zahlen (z.B. in den Tausendern) wird kaum jemand diese Spinner-Buttons sinnvoll für die Eingabe nutzen, da es „ewig“ dauert, allein durch Drücken des Pfeil-nach-oben-Buttons den Wert schrittweise von 0 auf z.B. 12993 zu erhöhen – da hat man den Wert schneller eingetippt. Dennoch sind die Spinner-Buttons ja „eigentlich“ unschädlich – bis auf die Tatsache, dass sie das Risiko erhöhen, einen bereits eingegebenen Wert später beim Scrollen durch die Seite *versehentlich* zu verändern, indem man eben unbeabsichtigt einen dieser Spinner-Buttons anklickt. Genau wenn Sie solche versehentlichen „Spins“ verhindern möchten, können Sie also auf Muster wie in 5. und 6. zurückgreifen.

Dieser Textfeld-Workaround hat aber auch wieder seine Nachteile: Bei den Zahlenfeldern wie in den Beispielen 1 bis 4 kann der Browser bei Fehleingaben nämlich sinnvolle sprechende Meldungen ausgeben wie in folgender Abbildung:



Number-Element mit Spinner-Buttons und sinnvoller Fehlermeldung bei Validierung

Wenn Sie dagegen ein Textfeld mit Pattern verwenden, zeigt der Browser bei Fehleingaben lediglich eine Meldung der Art »Halten Sie sich ans vorgegebene Muster!«. Was aber genau das vorgegebene Muster ist, sollte dann aus Ihrem Aufgabentext hervorgehen, andernfalls ist das nicht sehr hilfreich.

Daher würden wir im Normalfall die ersten Varianten mit Zahlenfeldern eher empfehlen.

Bei Beispiel 6. kommt als weiterer Nachteil hinzu, dass der Smartphone-Browser hier typischerweise eine Buchstabentastatur anzeigen wird, die erst manuell auf Zahleneingabe umgeschaltet werden muss.

Checkboxen als Pflichtfelder

In diesem Punkt besteht ein wesentlicher Unterschied zwischen der HTML-5-Validierung und der bisherigen JavaScript-Validierung aus Kapitel 2:

- Eine `<input type="checkbox" required … >` muss explizit angekreuzt werden, andernfalls kann nicht eingesendet werden. So etwas kennt man z.B. von Checkboxen zur verpflichtenden AGB-Zustimmung in Onlineshops. In Prüfungsaufgaben kann so etwas z.B. genutzt werden für Eigenständigkeitserklärungen (der Art „[ ] Ich versichere, diese Arbeit eigenständig bearbeitet zu haben.“).
- Falls Sie dagegen *mehrere Checkboxen mit gleichem `name`-Attribut* verwenden, wie es z.B. bei Multiple-Choice-Fragen üblich ist, und nun ausdrücken möchten, dass *mindestens eine* Checkbox aus dieser Gruppe markiert sein muss, so ist das mit reinem HTML nicht möglich! Dafür wäre in jedem Fall eine JavaScript-Lösung nötig. Die alte Javascript-Validation-Lösung aus Kapitel 2 leistet dies, wenn Sie eine (typischerweise die erste) der Checkboxen mit `class="required"` markieren.
  - Beispiel für die veraltete (Deprecated)-JavaScript-Lösung aus Kapitel 2:

    ```
    <form class="validated">
    <input type="checkbox" name="FeldA1" value="A" class="required">
    <input type="checkbox" name="FeldA1" value="B">
    </form>
    ```

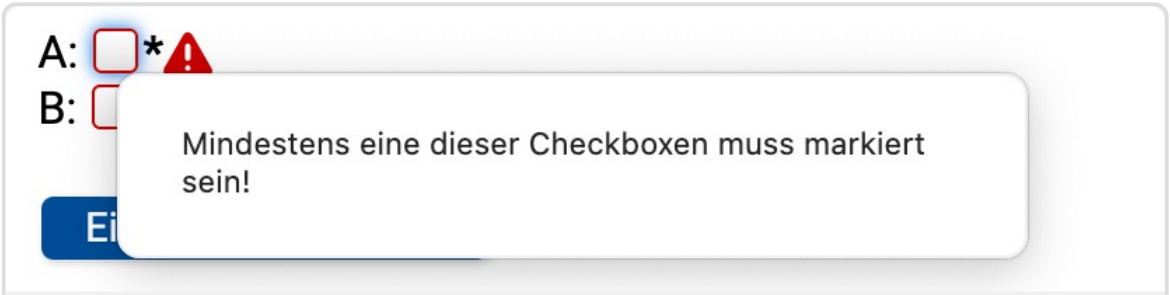
    In diesem Beispiel für die Anwendung der JavaScript-Validierung (siehe Kapitel 2) wird ausgedrückt, dass das Feld A1 ein Pflichtfeld ist, also mindestens eine der Checkboxen markiert werden muss (um eine der möglichen Eingaben „A“, „B“ oder „A,B“ zu erzeugen).
  - Im Gegensatz dazu würde der HTML5-Code

    ```
    <form class="validation-styles">
    <input type="checkbox" name="FeldA1" value="A" required>
    <input type="checkbox" name="FeldA1" value="B">
    </form>
    ```

    ausdrücken, dass die Checkbox A *in jedem Fall* angekreuzt werden muss, B dagegen optional ist. (Das ergibt für MC-artige Aufgaben in der Regel keinen Sinn!)
    Auch *beide* Checkboxen als `required` zu markieren, ergäbe keinen Sinn, weil dann auch *beide* zwangsläufig angekreuzt werden müssten, statt nur einer von beiden.
  - Um auch bei Nutzung der neuen HTML5-Validierung ausdrücken zu können, dass mindestens eine Checkbox einer Gruppe markiert werden soll, haben wir eine **neue Übungssystem-eigene JavaScript-Lösung** realisiert: Markieren Sie dazu (genau wie im ersten Beispiel) mindestens eine (typischerweise einfach nur die erste) der Checkboxen mit einer *Klasse* `class="required"`[2] und fügen Sie dem `form`-Element die Klasse `required-checkbox-groups` hinzu:

```
<form class="validation-styles required-checkbox-groups">
    <input type="checkbox" name="FeldA2" value="A" class="required"">
    <input type="checkbox" name="FeldA2" value="B">
</form>
```

(Diese neue Schreibweise unterscheidet sich also bis auf die Klassen am `form`-Tag nicht von der Syntax der alten JavaScript-Validierung.)
Dann wird das Online-Übungssystem per JavaScript immer *alle* Checkboxen einer Gruppe genau dann mit einem `required`-Attribut als Pflichtfelder markieren, wenn keine von ihnen markiert ist. Ist dagegen mindestens eine Checkbox markiert, wird das required-Attribut wieder von allen entfernt, damit eben nicht *alle* Checkboxen markiert werden müssen.
Außerdem ist dafür eine spezifische Meldung realisiert



Übungssystem-eigene Meldung für eine „Required-Checkboxgruppe“

Diese Problematik betrifft übrigens ausschließlich Checkboxen, nicht für **Radiobuttons**: Stellen wir obiges Beispiel auf Radiobuttons um:

```
<form class="validation-styles">
    <input type="radio" name="FeldA2" value="A" required>
    <input type="radio" name="FeldA2" value="B" required>
</form>
```

Der obige Beispielcode stellt bereits ganz ohne JavaScript-Nutzung schon im HTML-Standard sicher, dass *einer* der Radiobuttons dieser Gruppe markiert sein muss. (Die Markierung mehrerer davon wäre ja auch nicht möglich).

Auch hier würde es auch genügen, nur an *einem* dieser Inputs (z.B. dem ersten) das `required`-Attribut zu notieren, empfohlen wird aber, es in allen Inputs zu tun. Sofern Sie die Klasse `validation-styles` verwenden, macht das keinen Unterschied, da das Übungssystem im Zweifel fehlende `required`-Attribute an den restlichen Checkboxen nachträgt.

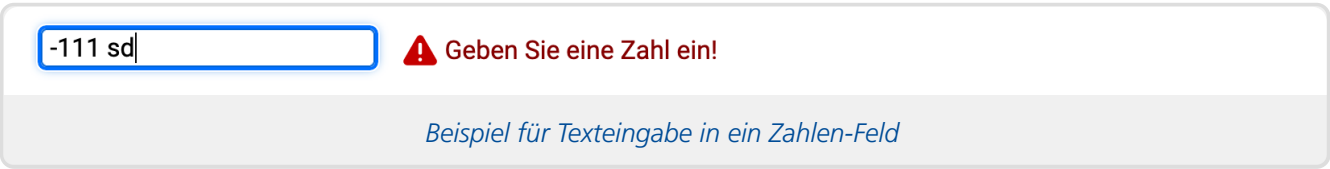
2. Formular-Eingabevalidierung per Java-Script (deprecated)

Das Online-Übungssystem bot seit 2011 die Möglichkeit, Formulare mit einer automatischen Prüfung der Eingaben per JavaScript auszustatten. Dies ist die ältere Technik, die weniger flexibel konfigurierbar ist, keinen Support für bestimmte virtuelle Smartphone-Tastaturen oder besondere Eingabeelemente wie Spin-Buttons an Zahlenfeldern bietet, und die auch nur funktioniert, sofern der Benutzer in seinem Browser JavaScript nicht abgeschaltet hat.

Dieses Verfahren wird für neu zu entwickelnde Aufgabenformulare nicht mehr empfohlen, steht aber nach wie vor zur Verfügung – wenn auch in erster Linie aus Gründen der Abwärtskompatibilität (damit bestehende Aufgaben, die es nutzen, weiterhin funktionieren und nicht angepasst werden müssen). Wir behalten uns dennoch vor, das Verfahren zukünftig einzustellen, falls es irgendwann nicht mehr ohne Codeanpassungen mit neueren Browsern kompatibel sein sollte.

Um diese JavaScript-Formularvalidierung zu nutzen, müssen Sie selbst keinen JavaScript-Code schreiben und auch keine Scriptdateien einbinden, sondern lediglich Ihren HTML-Code mit bestimmten `class`-Attributen versehen.

Dazu kann für einzelne Formularfelder festgelegt werden, ob sie z.B. Pflichtfelder sein sollen (d.h. nicht leer gelassen werden dürfen), ob sie nur Ziffern oder Zahlen (mit Fließkommadarstellung) oder ein Datum aufnehmen dürfen. So lange diese Bedingungen nicht sämtlich erfüllt sind, lässt sich das Formular nicht absenden.



Beispiel für Texteingabe in ein Zahlen-Feld

Obige Abbildung zeigt beispielhaft eine Fehlermeldung für ein als numerisch markiertes Feld. Derartige Prüfungen vor dem Einsenden können (insbesondere bei automatisch nach Einsendeschluss bewerteten Aufgaben) Studierenden helfen, fehlerhafte Eingaben schon vor dem Einsenden (und nicht erst nach Erhalt der Korrektur) zu erkennen.

Einrichtung

Um für ein HTML-Formular die Eingabevalidierung zu aktivieren, fügen Sie dem Form-Tag die Klasse `validated` hinzu (`<form … class="validated" …>…</form>`). Weiterhin sind zu jedem Formularelement, für das eine Validierung stattfinden soll, die jeweiligen Validierungsregeln ebenfalls als Klasse (d.h. im class-Attribut) anzugeben. Insb. folgende Klassen sind möglich:

Klasse	zulässige Eingaben
`required`	Pflichtfeld: Das Feld darf nicht leer gelassen werden. Die Klasse `required`lässt sich mit den folgenden Klassen kombinieren:
`digits`	Eine Ziffernfolge (ohne weitere Zeichen wie Plus, Minus, Komma oder Punkt)
`integer`	Eine Ziffernfolge optional mit Vorzeichen + oder - davor, aber ohne Dezimalstellen.
`number`	Eine Zahl, die wahlweise mit Minus beginnen darf und entweder einen Punkt oder ein Komma als Dezimaltrenner (jedoch keinen Tausendertrenner) aufweisen darf. Beispiele für korrekte Zahlen: „5“, „5,9“ oder „-456.99“. Ab sofort wird auch die wissenschaftliche Exponentenschreibweise unterstützt, z.B. „1,3e9“ für $\mathbf{1,3 \cdot 10^9}$ oder „0.2E-14“ für $\mathbf{0,2 \cdot 10^{-14}}$. Das E darf groß oder klein geschrieben werden, und einem positiven Exponenten darf, muss aber kein Vorzeichen (+) vorangestellt werden.
`dateDE`	Datum im Format `tt.mm.jjjj` (also zweistelliger Tag, zweistelliger Monat, vierstelliges Jahr)
`dateISO`	Datum im Format `jjjj/(m)m/(t)t` (vierstelliges Jahr, ein- oder zweistelliger Monat, ein- oder zweistelliger Tag, alternativ zu ‚/‘ ist auch ‚-‘ als Trennzeichen erlaubt)
`url`	Absolute URL (muss mit `http://`, `https://` oder `ftp://` beginnen, andere URLs werden nicht erkannt)
`email`	E-Mail-Adresse

Beispiel für ein validiertes Formular mit einem Pflichtfeld zur Eingabe einer Zahl:

```
<form class="validated"
action="$WebAssignServer/$Veranstaltername/Einsendung/$KursNr/$VersionsNr/$Aufg
abenheftNr/$AufgabenNr" method="post">
    …
    <input type="text" class="number required" name="FeldA1">
    …
</form>
```

Hinweise

- Die Klasse `number` lässt explizit sowohl ein Komma als auch einen Punkt als Eingabe zu. Dies wurde in Übereinstimmung mit dem Aufgabenbewerter (dem im Übungssystem vorinstallierten Korrekturmodul für automatisch bewertete Aufgaben) gewählt: Bei automatisch bewerteten Fragen nach Zahlen in einem Intervall werden ebenfalls sowohl Komma als auch Punkt als Dezimaltrenner akzeptiert. Der Aufgabenerstellungsassistent zur Erstellung automatisch bewerteter Aufgaben im Online-Übungssystem fügt zukünftig diese Formularvalidierung bei Fragen nach Zahlen automatisch ein. Bei bereits bestehenden automatisch bewerteten Zahlen-Fragen lässt sich die Validierung durch Nachtrag der `class`-Attribute wie oben beschrieben nachrüsten.
- Die Klasse `required` ist zumindest im Kontext von Aufgabenseiten mit Vorsicht zu genießen: Normalerweise ist es erstrebenswert, dass ein Studierender auch ein unvollständig ausgefülltes Aufgabenformular einsenden kann, um seine (Zwischen-)ergebnisse zu speichern. Er kann dann später die Aufgabenseite erneut aufrufen und weitere Eingaben ergänzen. Mit der Angabe `required` verhindern Sie ein solches Zwischenspeichern, so lange nicht alle Pflichtfelder ausgefüllt sind! (Durch Abschalten von JavaScript im Browser lässt sich diese Sperre jedoch umgehen.)
- Die `required`-Klasse kann nicht nur Textfeldern, sondern auch anderen Formularelementen

wie z.B. Checkboxen zugeordnet werden. So ist es z.B. möglich, eine Checkbox der Art „☐ Ich versichere, dass …“ einzufügen und ein Einsenden erst zu erlauben, sobald die Checkbox angekreuzt wurde:

```
<input type="checkbox" class="required" name="FeldA2" value="Ja">
```

- Auch auf ganze Gruppen von RadioButtons oder Checkboxen (d.h. mehrere Input-Elemente mit gleichem `name`-Attribut) kann die `required`-Klasse angewendet werden. Es genügt dabei, sie dem *ersten* Input-Element der Gruppe zuzuordnen. Die Semantik ist dann *nicht* etwa, dass genau die mit dem `class="required"`-Attribut markierte Checkbox (bzw. Radiobutton) markiert werden muss, sondern genau ein Radiobutton bzw. mindestens eine Checkbox der Gruppe. Beispiel:

  ```
  <input type="radiobutton" name="FeldA3" value="Ja" id="radJa"
  class="required"> <label for="radJa">Ja</label><br>
  <input type="radiobutton" name="FeldA3" value="Nein" id="radNein"> <label
  for="radNein">Nein</label>
  ```

  Das Beispiel zeigt zwei Radiobuttons einer Gruppe, über die ein Student wahlweise "Ja" oder "Nein" antworten könnte. Die `required`-Klasse bewirkt, dass die Einsendung erst abgeschickt und gespeichert werden kann, wenn der Student hier eine Auswahl getroffen hat. Ein Einsenden ohne Markieren einer der beiden Optionen wird verhindern (sofern JavaScript im Browser nicht deaktiviert wurde).

- Soll eine Select-Box (Auswahlliste) per required als Pflichtfeld markiert werden und eine erste Auswahloption (z.B. mit dem Text „Bitte auswählen…“) enthalten, die als „leere Auswahl“, also als nicht ausgefülltes Pflichtfeld interpretiert werden soll, so ist dieser „leeren“ Auswahloption das Attribut value="" zuzuweisen:

  ```
  <select class="required" name="FeldA3">
      <option value="">Bitte auswählen...</option>
      …
  </select>
  ```

Fußnoten

1. Beachten Sie, dass dabei dann auch die jQuery-Library eingebunden wird, die wiederum ihr globales `jQuery`-Objekt auch unter der Alias-Variable `$` ablegt. Falls Sie selbst ein anderes JavaScript-Framework wie Prototype o.ä. verwenden, das ebenfalls die Variable `$` definiert, könnte es (je nach Reihenfolge der Einbindung beider Libraries) also dazu kommen, dass entweder Ihr Scriptcode oder die Formularvalidierung nicht korrekt funktioniert.
2. Für diese Lösung wurde bewusst der Einsatz einer *Klasse* `class="required"` an Stelle des HTML-Attributs `required` gewählt: Falls im Client-Browser JavaScript inaktiv sein sollte, wird so zwar das Pflichtfeld-Constraint nicht sichergestellt, aber wenigstens ist jede gültige Eingabe (Kombination von Checkboxen) möglich. Hätte man statt dessen im HTML einfach ein `required`-Attribut mit ganz anderer Semantik eingesetzt und verließe sich darauf, dass dessen Semantik nachträglich per JavaScript modifiziert wird, so wäre die Aufgabe mit abgeschaltetem JavaScript nicht mehr korrekt bearbeitbar.