online Übungssystem

WYSIWYG-Editoren, In-Browser-Korrektur, Syntax Highlighting

Eingebettete Rich-Text-Editoren, In-Browser-Korrektur und Syntax Highlighting im Online-Übungssystem

Autor:
Immo Schulz-Gerlach, ZMI, FeU-Anwendungen

Version:
3.2 vom 13.01.2023

FernUniversität in Hagen

Inhaltsverzeichnis

Einleitung / Motivation

Eine häufig genutzte Variante von Einsendeaufgaben im Online-Übungssystem sieht vor, dass Studenten ihre Lösungen in Form mehrzeiligen Klartextes in Eingabefelder eingeben, die direkt in der Aufgabenseite zur Verfügung stehen, und dass solche Einsendungen später von Korrekturkräften manuell korrigiert werden. Im Normalfall besteht ein derartiges Eingabefeld innerhalb der Aufgabenseite aus einer einfachen Textbox (HTML-Textarea), in die lediglich *unformatierter* Text eingegeben werden kann. Ein Aufgabenautor hatte zwar schon immer die Möglichkeit, einen JavaScript-HTML-Editor zur Eingabe (HTML-)*formatierten* Textes in die Aufgabenseiten einzubetten, nur musste er sich bisher für ein Produkt entscheiden, die benötigten Dateien als Kursressourcen hochladen und sich selbst eingehend mit der Einbindung des Editors in den HTML-Quelltext sowie dessen Konfiguration beschäftigen.

Das zweite Kapitel dieses Dokuments (nach dieser Einleitung) geht daher auf das (seit Dezember 2010 im Online-Übungssystem verfügbare) Feature zur Einbindung eines **integrierten Editors für formatierten Text** in eine Aufgabenseite ein, die sehr einfach lediglich durch Aufnahme einer einzigen Kommentarzeile erfolgen kann. Das Online-Übungssystem stellt dazu mehrere alternative vom ZDI vorkonfigurierte Editor-Ausprägungen zur Verfügung.

Das dritte Kapitel dieses Dokuments beschreibt die (ebenfalls seit Dezember 2010 verfügbare) sog. „**In-Browser-Korrektur**".
Das „*herkömmliche* HTML-Korrekturverfahren" sah vor, dass eine Korrekturkraft eine Suite aus Browser und HTML-Editor, wie z.B. »Mozilla SeaMonkey«, installiert. Zur Korrektur wählt der Korrektor dann im Browser eine zu korrigierende Einsendung aus, die ihm daraufhin im Browser angezeigt wird. Mit einem kurzen Befehl (Strg-E in SeaMonkey) öffnet er dann die Korrektur im Editor, kann sie beliebig bearbeiten und anschließend über einen Publish-Befehl wieder im Online-Übungssystem speichern. Über ein Formular im Browser trägt der Korrektor anschließend noch die vergebene Punktzahl ein und vermerkt den Korrekturstatus.

Diese „herkömmliche HTML-Korrektur" steht zwar – insbesondere aus Abwärtskompatibilitäts- und Flexibilitätsgründen – unverändert weiter zur Verfügung, der Einsatz wird jedoch nicht mehr empfohlen. Als Ersatz gibt es nun mit der „In-Browser-Korrektur" eine Alternative mit insb. folgenden Eigenschaften:

- Der Korrektor korrigiert direkt im Browser, er benötigt keine spezielle Editor-Software.
- Der Korrektor kann im Regelfall nicht mehr die gesamte Einsendung modifizieren, sondern bekommt in der Korrekturseite Textboxen zur Eingabe seiner Kommentare zur Verfügung gestellt.
- Die Kommentare des Korrektors werden in der dem Studenten angezeigten Korrektur automatisch optisch hervorgehoben (standardmäßig durch rote Schriftfarbe), so dass der Korrektor seine Eingaben nicht mehr manuell hervorheben muss.
- Auch für die In-Browser-Korrektur (nicht nur für studentische Einsendungen) kann der bereits erwähnte integrierte HTML-Editor verwendet werden.

Der Vollständigkeit halber sei erwähnt, dass es mit der „rein dateibasierten Korrektur" noch ein drittes Verfahren neben In-Browser-Korrektur und herkömmlicher HTML-Korrektur gibt, nämlich eine Korrektur ganz ohne Bearbeitung einer HTML-Korrekturseite: Wenn die Studierenden lediglich Dateien, z.B.im PDF-Format, einsenden und die Korrektur ausschließlich in Form des Uploads von Dateien (z.B. annotierte Versionen der Dateieinsendungen und/oder Uploads ausgefüllter Korrekturformulare im PDF-Format) stattfinden soll, genügt es, wenn die vom Online-Übungssystem erzeugten Korrekturseiten nur noch Links zu den hochgeladenen einzelnen Dateien enthalten und demzufolge gar nicht mehr von den Korrektor:innen bearbeitet werden müssen – weder per In-Browser-Korrektur noch auf die „herkömmliche Art". Das ist aber kein Thema dieses Handbuchs. Siehe Handbuch für Aufgabenautoren I / <https://online-uebungssystem.fernuni-hagen.de/download/Aufgabenerstellung/Aufgabenerstellung.html#hb> für eine genauere Gegenüberstellung der Korrekturmodi.

Das vierte Kapitel richtet sich speziell an Anbieter von Aufgaben, in denen Studenten Quelltexte (z.B. von Programmen, Webseiten oder Datenbankanfragen) einsenden müssen oder in denen solche in den Aufgabentexten oder Musterlösungen dargestellt werden sollen. Es stellt ein seit Mai 2014 verfügbares Feature des Online-Übungssystems vor: **Automatische Syntaxhervorhebung und Quelltext-Einsendungen**.

Eingebettete WYSIWYG-HTML-Editoren

Seit Dezember 2010 ist der in JavaScript implementierte Open-Source-Editor *TinyMCE* ins Online-Übungssystem integriert. D.h. die benötigten Ressourcen (JavaScript-Quellen, Grafiken, Konfigurationsdateien) sind zentral auf dem Übungssystem-Server hinterlegt, und das Übungssystem bietet Aufgabenautoren verschiedene vorkonfigurierte Ausprägungen mit unterschiedlich großer Funktionsvielfalt zur Einbindung in Aufgabenseiten (für studentische Eingaben) oder Korrekturseiten (für In-Browser-Korrektur) an. Anfangs wurde Version 3 eingesetzt, im Mai 2014 wurde auf die grundlegend überarbeitete Version *TinyMCE 4* umgestellt, und im Januar 2023 auf *TinyMCE 5*. Im Standardfall wird immer der gerade aktuelle / neueste installierte Editor genutzt. Die beiden Vorversion sind aber parallel weiterhin installiert und können zumindest in Aufgabenseiten und für die In-Browser-Korrektur noch alternativ verwendet werden, wenn vom Aufgabenautor explizit Version 3 oder 4 angefordert werden.

Die nachfolgenden Abbildungen *in diesem Kapitel* zeigen jeweils TinyMCE 5 in der im Januar 2023 installierten Version. In späteren Kapiteln, die sich nicht primär mit der WYISWYG-Einbindung beschäftigen, können auch noch ältere Abbildungen vorhanden sein, die frühere TinyMCE-Versionen zeigen.

Editor-Typen

Einfacher Editor

Die einfachste Ausprägung ist ein einfacher Editor, der lediglich Grundfunktionen zur Zeichenformatierung (wie z.B. Fettdruck) besitzt, Aufzählungen ermöglicht und eine Undo- und Redo-Funktionalität aufweist:

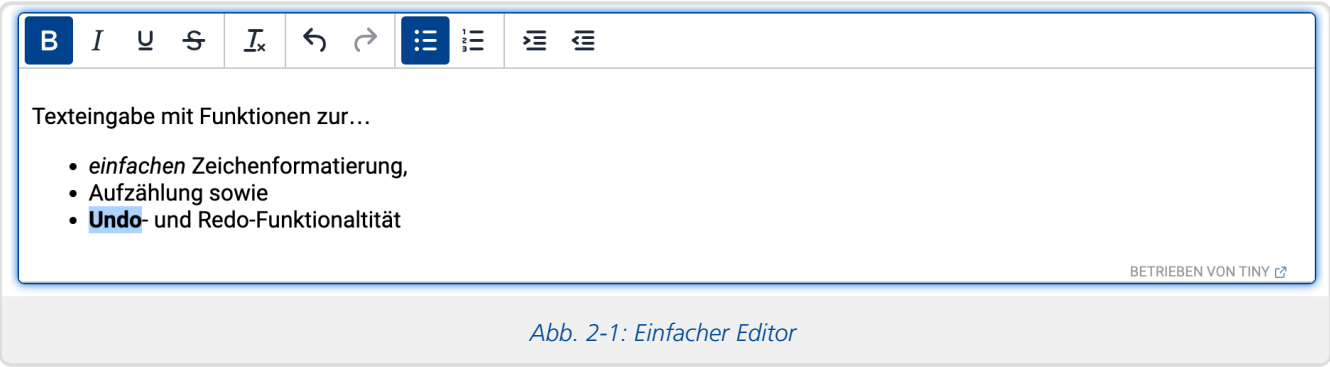


Abb. 2-1: Einfacher Editor

Weiterhin unterstützt bereits der einfache Editor das Zerlegen des Textes in Absätze. Dazu wird bei Betätigen der Enter-Taste ein neuer Absatz begonnen, während Shift-Enter einen Zeilenumbruch innerhalb des Absatzes einfügt.

Der einfache Editor ist besonders geeignet für kurze Texte, die nicht in großem Stil formatiert oder gegliedert werden sollen, sondern als kurze Fragmente in ein größeres Dokument (Einsendung bzw. Korrektur) eingebunden werden und für die lediglich Basisfunktionalitäten zur einfachen Hervorhebung / Betonung einzelner Wörter sowie ggf. die Eingabe von Stichpunkten gewünscht sind. Insbesondere für Korrektor-Kommentare im Rahmen der In-Browser-Korrektur (vgl. nachfolgenden Kapitel) bietet sich der einfache Modus an.

Erweiterter Editor

Als anderes Extrem neben dem einfachen Editor steht ein erweiterter Editor zur Verfügung, der viele zusätzliche Möglichkeiten der Zeichenformatierung (z.B. Schriftfarben) und erweiterte Funktionen (z.B. Suchen und Ersetzen, Drucken, Vollbild, Sonderzeichenpalette, Tabellen u.v.m.) bietet. Im »Erweitert«-Menü (ebenso wie in der Menüleiste und dem Kontextmenü) finden sich außerdem zwei FernUni-Spezifische Funktionen zum Einfügen von

1. Programmquelltexten (mit Syntax-Highlighting, siehe auch das dritte Kapitel) und

2. mathematischen Formeln[1] (siehe auch separates Handbuch zu Formelsatz <https://online-uebungssystem.fernuni-hagen.de/download/TeXIntegration/TeXIntegration.html>).

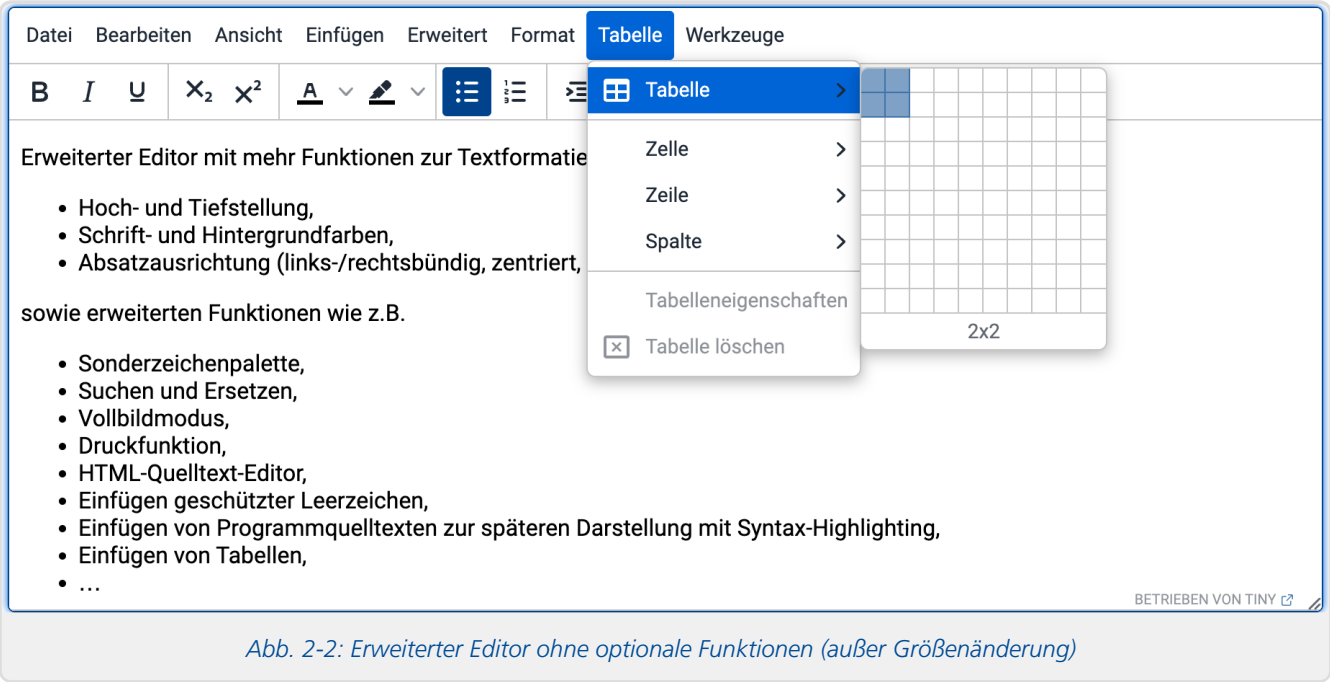


Abb. 2-2: Erweiterter Editor ohne optionale Funktionen (außer Größenänderung)

In obiger Abbildung ist der erweiterte Editor beinahe in Grundeinstellung abgebildet. Als einzige Zusatzoption wurde die manuelle Größenänderung aktiviert: Nutzer:innen können über den Greifer unten rechts mit der Maus die Größe des Editors verändern.

Hinweis: Die manuelle Größenanpassung funktioniert nur mit Maussteuerung, nicht auf Multitouch-Screens. Auf Mobilgeräten wie Smartphones lädt der Editor eine abweichende Darstellung, die daher auch insbesondere auf diesen Greifer verzichtet.

Der erweiterte Editor mit Funktionen wie z.B. einer Vollbildansicht oder Suchen und Ersetzen ist insbesondere für größere „Dokumente“ ausgelegt. Falls es gewünscht wird, dass der Nutzer (Student bzw. Korrektor) die Möglichkeit haben soll, seinen Text durch Überschriften verschiedener Ordnung zu strukturieren, kann optional eine Auswahl entsprechender Absatzformate aktiviert werden. Unabhängig davon ist es auch möglich, die Auswahl einer Schriftart oder einer Schriftgröße zu erlauben[2]. Man sollte sich aber überlegen, ob das wirklich sinnvoll ist, oder ob z.B. unnötiger Einsatz von Schriftarten und Schriftgrößen in gewissen studentischen Einsendungen nicht vielmehr die Les- und Korrigierbarkeit beeinträchtigen könnte.

Die nachfolgende Abbildung zeigt den erweiterten Editor mit allen drei optionalen Auswahlmöglichkeiten freigeschaltet. Außerdem wurde hier eine Statusleiste am unteren Rand eingebunden, die einerseits für Personen mit HTML-Kenntnis die DOM-Struktur darstellt und außerdem (seit Umstellung auf TinyMCE 5 im Januar 2023) auch einen Wort- oder Zeichenzähler enthält (Wechsel zwischen Wort- und Zeichenzähler durch Anklicken möglich). Die Wort- und Zeichenzählfunktion ist auch (selbst bei abgeschalteter Statusleiste) übers Menü erreichbar (und dort auch ausführlicher), aber nur in der Statusleiste ist einer der Werte stets im Blick.

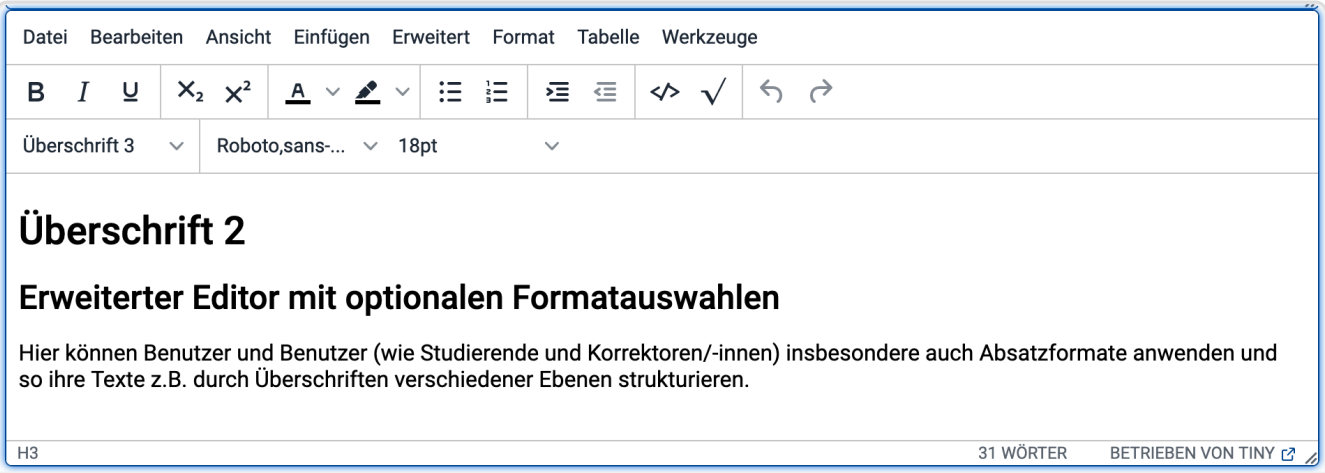


Abb. 2-3: Erweiterter Editor mit optionalen Format-Auswahlmöglichkeiten

Der erweiterte Editor unterstützt insbesondere auch das Einfügen von Grafiken aus der Zwischenablage (ggf. nicht in jedem Browser). Diese Bilddaten werden nicht – wie normalerweise in Webseiten üblich – als Dateien hochgeladen und dann im Dokument „nur" verlinkt, sondern werden direkt als Binärdaten[3] mit in den HTML-Code eingefügt. Vorteil: Studenten können in mit erweitertem Editor ausgestatteten Texteingabefeldern irgendwo in ihrem Text Bilder einbetten, ohne dass Übungssystem oder Aufgabenautor bestimmte Vorkehrungen wie spezielle Dateiupload-Felder in der Aufgabe vorsehen müssten. Nachteile sind vor allem die Dateigröße (eine solche eingebettete Grafik benötigt innerhalb des HTML-Quelltextes etwa ein Drittel *mehr* Speicherplatz als in einer lokal gespeicherten Grafikdatei!) und die Einschränkung auf vom Browser darstellbare Grafikformate (i.W. JPEG, PNG und GIF – insbesondere können keine PDF-Grafiken eingebettet werden).

Wenn die Einsendung von Grafik- oder PDF-Dateien zu einer Aufgabe die Regel ist / erwartet wird, sollten Sie als Aufgabenautor dazu normalerweise besser ein Datei-Einsendefeld in der Aufgabe vorsehen und nicht auf diese Grafik-Einfügefunktion des Editors verweisen! Aber wenn Sie eigentlich nur Texteinsendungen erwarten, ein Student aber meint, seine Ausführungen besser anhand einer kleinen Grafik erläutern zu können, so hat er hierzu im erweiterten Editor standardmäßig die Möglichkeit. Sie können als Aufgabenautor aber auch für den erweiterten Editor die Möglichkeit, Bilder einzubetten, ganz abschalten (siehe Einbindung).

Darstellung auf Smartphones

Auf Mobilgeräten wie Smartphones wird der erweiterte Editor (seit Umstellung auf TinyMCE5 / Januar 2023) in einer kompakteren Darstellung geladen:

- Die Menüzeile wurde zur Platzersparnis verkürzt, wobei aber alle aus dem Menü entfallenen Funktionen in der Symbolleiste verfügbar sind. Dazu enthält die Symbolleiste ein paar zusätzliche Symbole gegenüber der Normaldarstellung.
- Da die Symbolleiste nun mehr Inhalte enthält als auf dem kleinen Raum darstellbar sind, ist sie horizontal scrollbar (was auch Touchscreens ja einfach und komfortabel möglich ist, im Gegensatz zu Maussteuerung, die meist horizontales Scrollen nur über Ziehen von Scrollbalken unterstützt).
- Die Auswahlboxen für Absatzformate oder Schriftart und -Größe werden, sofern überhaupt aktiviert, auch nicht mehr in einer eigenen Leiste dargestellt, sondern mit in die nun einzige Symbolleiste mit aufgenommen (hinter den Befehlssymbolen „angehängt").

Wie oben bereits erwähnt, entfällt im „Mobil-Modus" ebenfalls der „Greifer" zur manuellen Größenänderung entfällt, weil diese eine Maus erfordert und auf Touch Screens nicht unterstützt wird.

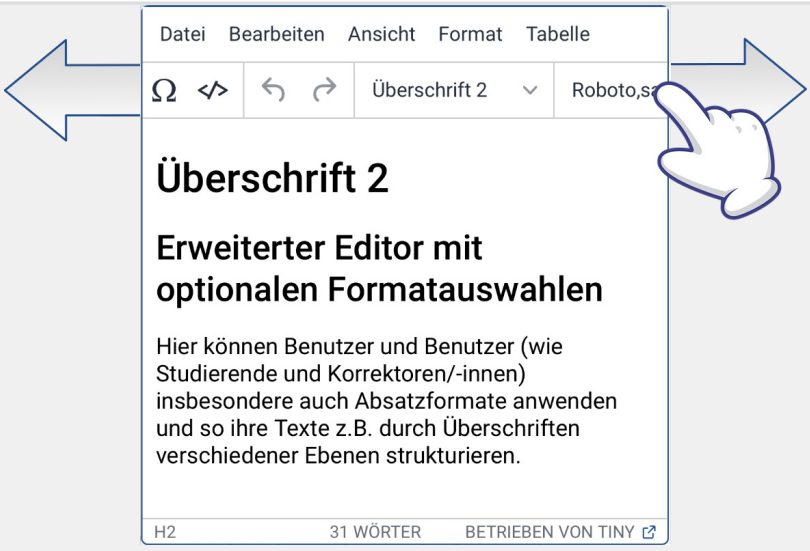


Abb. 2-4: Erweiterter Editor auf Smartphone

„Kompakter" Editor

Als Kompromiss-Lösung zwischen dem einfachen und dem erweiterten Editor bietet das Online-Übungssystem eine weitere Editor-Variante, die im Funktionsumfang über den einfachen Editor hinausgeht und einen ausgewählten Teil der Funktionen des erweiterten Editors mitbringt, dabei jedoch mit einer einzigen Symbolleiste auskommt. Dieser „kompakte" Editor ist für die Einbindung in Seiten gedacht, in denen viele Eingabefelder vorhanden sind und der erweiterte Editor „zu dick aufträgt", der einfache Editor jedoch zu wenig Funktionalität bietet:

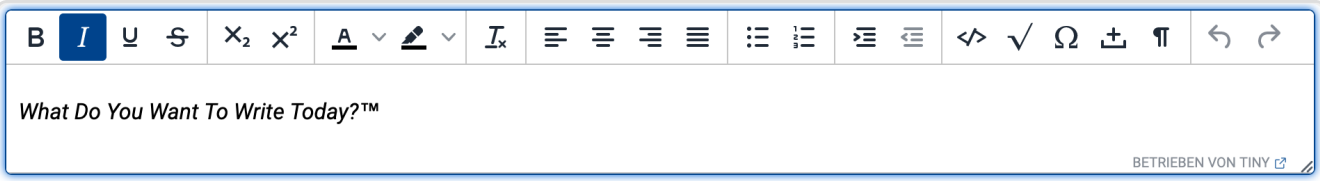


Abb. 2-5: Kompakter Editor

Editor-Größen & Größenanpassungen

Sie können zunächst für jede Texteingabebox (egal, ob diese mit WYSIWYG-Editor ausgestattet werden soll oder nicht) eine Vorgabegröße wählen, sie im Standardfall beim Laden der Seite angezeigt werden sollen.

Dazu gibt es aber noch Möglichkeiten der manuellen und der automatischen Anpassung der Größe: Die Editorbreite z.B. kann sich auf Wunsch immer an die Seitenbreite (bzw. genauer die Breite der Textspalte) anpassen. Die Editorhöhe kann sich auf Wunsch an den Editor-Inhalt anpassen, also „mit dem Inhalt mitwachsen".

Weiterhin können Editoren (außer dem einfachen Typ) eine Möglichkeit zur manuellen Größenänderung per Maus bieten: Der kompakte Editor hat immer den Greifer zur Größenänderung rechts unten (außer auf Mobilgeräten), beim erweiterten Editor können Sie das selbst einstellen.

Automatische Höhenanpassung an Inhalt

Seit Januar 2023 können Sie optional die automatische Höhenanpassung verwendet. Diese bewirkt, dass der Editor in der Höhe mit seinem Inhalt wachsen bzw. schrumpfen kann. Für eine noch leere Textbox wird z.B. ein sehr kleiner Editor angezeigt. Füllt man ihn mit Text, wächst er in der Höhe immer weiter, so dass möglichst immer der gesamte Text sichtbar ist und kein Scrollbalken innerhalb des Editors nötig wird.

Das „automatische Wachstum“ ist jedoch beschränkt auf etwas weniger als die Fensterhöhe, so dass immer die gesamte Editorbox vo,n der Menü- bzw. Symbolleiste bis zur Fußleiste (wenn eingeschaltet) komplett sichtbar ist. Für noch längere Eingaben wird dann doch wieder auf einen Scrollbalken innerhalb des Editors zurückgegriffen.

Ob solche mitwachsenden Editoren in Ihren Aufgabenseiten (bzw. Korrekturseiten für die In-Browser-Korrektur) sinnvoll sind oder doch feste Ausgangsgrößen, ggf. mit Möglichkeit zur manuellen Anpassung zu bevorzugen sind, hängt sicher auch ein wenig von Art und Aufbau der Aufgabenseiten ab. Daher wird diese Entscheidung auch nicht vom ZDI vorgegeben, sondern Sie können selbst dazwischen wählen.

Für die automatische Größenanpassung spricht z.B., dass gerade beim Laden einer Aufgabenseite mit mehreren Textboxen, die anfangs noch leer sind, der geringe Platzbedarf der kleinen Editorboxen einen besseren Gesamtüberblick ermöglicht. Beim Aufruf einer Seite mit längeren Texteingaben spricht dafür, dass diese besser am Stück lesbar sind und nicht in jeder einzelnen Textbox einzeln gescrollt (oder der Editor manuell vergrößert) werden muss. Darunter leidet dann aber eventuell der Überblick über die gesamte Seite etwas.

Die automatische Größenanpassung steht für alle Editortypen (einfach, kompakt, erweitert) zur Verfügung. Im Übrigen ist sie auch mit der manuellen Größenanpassung (im kompakten Editor immer aktiv, im erweiterten zuschaltbar) kombinierbar: In dem Fall können die Nutzer z.B. einen ihnen zu großen Editor manuell wieder verkleinern. Manuelle Größenänderungen sollten bestehen bleiben, so lange der Text nicht bearbeitet wird. Bei der Bearbeitung kann jederzeit wieder eine automatische Anpassung der Höhe erfolgen.

Anpassung an Seitenbreite

Wenn Sie die Breite der Textbox nicht selbst auf eine konstante Größe festlegen möchten, können Sie die automatische Breitenanpassung an die Absatzbreite der Seite verwenden (empfohlen).

Editoren mit der Möglichkeit zur manuellen Größenänderung sind dann (zumindest in den neueren TinyMCE-Versionen) auf die Breite fixiert, d.h. über den Greifer rechts unten – sofern vorhanden – können Nutzer:innen dann nur noch die Editorhöhe nachjustieren.

Einbindung in eine Aufgabenseite oder Korrektur

Assistentengestützte Einbindung

Falls Sie die Aufgabe mit einem Assistenten des Online-Übungssystems erzeugen, fügen Sie Ihrer handbewerteten Aufgabe einfach ein „Freitext-Eingabebox“-Element hinzu und stellen Sie ein, ob und ggf. welchen Editortyp Sie verwenden wollen.

Dort haben Sie dann auch weitere Einstellmöglichkeiten wie insb.

- eine Option zur automatischen Anpassung an die Seitenbreite (für alle Editoren, nicht nur für WYSIWYG, daher ganz oben bei den Grundeinstellungen zur Größe zu finden, sowie
- eine Option zur automatischen Höhenanpassung, sofern Sie einen WYSIWYG-Editortyp ausgewählt haben.
- Bei Auswahl des erweiterten Editortyps werden entsprechend weitere Einstellmöglichkeiten eingeblendet.

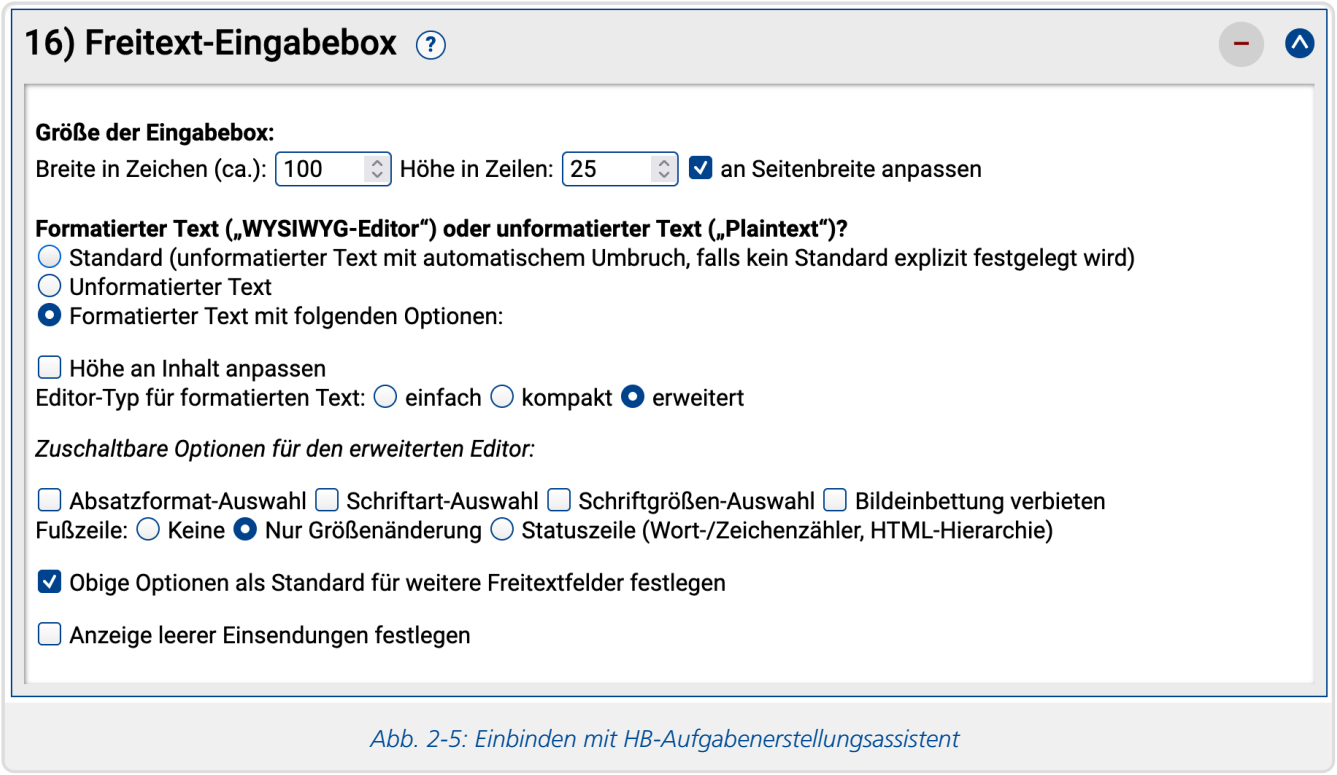


Abb. 2-5: Einbinden mit HB-Aufgabenerstellungsassistent

Manuelle Einbindung in HTML-Code

Zur Einbindung des Editors in einer der oben abgebildeten Varianten ist lediglich das Einfügen einer kurzen HTML-Kommentar-Zeile in das jeweilige HTML-Dokument (Aufgabenseite bzw. Korrekturseite) nötig, wahlweise im HEAD oder im BODY-Bereich des Dokuments.

Zur Einbindung des **einfachen Editors** genügt der Kommentar `<!-- WYSIWYG -->`. Dieser bewirkt, dass zur Laufzeit – sofern JavaScript im Browser aktiviert ist – jede Textarea der Seite durch einen einfachen HTML-Editor wie in Abbildung 2-1 gezeigt ersetzt wird.

Zur Einbindung des **kompakten Editors** dient der Kommentar `<!-- WYSIWYG kompakt -->`.

Zur Einbindung des **erweiterten Editors** ist dem Kommentar das Wort `erweitert` hinzuzufügen. Die optionalen Format-Auswahllisten des erweiterten Editors werden durch Hinzufügen der Wörter `Absatzformate`, `Schriftarten` und/oder `Schriftgroessen` aktiviert.
`<!-- WYSIWYG erweitert -->` erzeugt also z.B. den erweiterten Editor ganz ohne Zusätze, während `<!-- WYSIWYG erweitert Absatzformate Schriftarten -->` den erweiterten Editor mit Absatzformat-Auswahl und Schriftart-Auswahl, jedoch ohne Möglichkeit zur Schriftgrößenmodifikation einbindet.

Um den Greifer zur Größenänderung der Eingabebox zu aktivieren, ist weiterhin das Wort `groessenaenderung` zum Kommentar hinzuzufügen.
Alternativ das Wort `statusleiste` an Stelle von `groessenaenderung` angegeben werden, dann enthält die Fußleiste neben dem Greifer zur Größenänderung auch noch (wie [oben][erweitertereditor] bereits beschrieben und abgebildet) einerseits einen Wort- bzw. Zeichenzähler[4] (umschaltbar) als auch einen Pfad im DOM-Baum zur aktuellen Auswahl anzeigt (z.B. „UL » LI“, falls der Cursor gerade innerhalb eines Listenelementes einer unnummerierten Liste steht). Letzteres ist für Benutzer mit HTML-Kenntnissen mitunter hilfreich.

Hinweis: Seit Umstellung auf TinyMCE 5 ist es auch möglich, eine *automatische* Größenanpassung zu aktivieren (siehe folgenden Unterabschnitt). Die manuelle Größenänderung kann aber dennoch zusätzlich angeboten werden.

Falls Sie das im erweiterten Editor standardmäßig erlaubte Einfügen von Bildern aus der Zwischenablage (Bilder eingebettet in HTML-Code, Base64-kodiert, siehe Abschnitt Editor-Typen) *nicht erlauben* wollen, fügen Sie dem

Kommentar zusätzlich das Wort `keinebilder` hinzu[5].

Größe der Texteditoren, Optionen zur Größenanpassung

Bei den oben beschrieben Einbindungen haben die Editoren stets eine feste Ausgangsgröße, die in etwa dem Platzbedarf der jeweils von ihnen zu ersetzenden Textareas entspricht, sich also an den Attributen `rows` und `cols` des `<textarea...>`-Tags orientiert. Für den Editor-Inhalt bleibt dabei Raum für weniger sichtbare Textzeilen als in `cols` angegeben, da ein Teil dieser Textarea-Fläche mit Symbol- sowie ggf. Menü- und Statusleisten belegt wird.

Der einfache Editor hat immer fest diese Größe, der kompakte Editor hat (außer auf Touch Screens) immer die Möglichkeit zur manuellen Größenänderung, beim erweiterten Editor können Sie letztere selbst aktivieren oder weglassen.

Darüber hinaus haben Sie noch folgende Möglichkeiten (siehe auch einleitenden Abschnitt Editor-Größen & Größenanpassungen):

Automatische Höhenanpassung an Inhalt

Um die automatische Größenanpassung (Höhenanpassung) zu aktivieren, fügen Sie dem Kommentar das Schlüsselwort `autogroesse` hinzu, z.B.:

```
<!-- WYSIWYG kompakt autogroesse -->
```

Die automatische Größenanpassung ist, wie oben schon gesagt, auch mit manueller Größenanpassung kombinierbar, z.B.:

```
<!-- WYSIWYG erweitert absatzformate groessenaenderung autogroesse -->
```

Die `rows`-Angabe der Textarea wird dann komplett ignoriert (zumindest nach erfolgreichem Laden des WYSIWYG-Editors, eine Angabe schadet jedoch nicht, nur für den Fallback-Fall.)

Anpassung an Seitenbreite

Normalerweise hat eine Textarea eine Breite, die sich ungefähr an der Spaltenzahl aus dem `cols`-Attribut orientiert, und auch die WYSIWYG-Editoren übernehmen diese Breite.

Sie können jedoch der jeder Textarea in einer Übungssystem-Aufgabenseite im `class`-Attribut die Klasse `fullwidth` zuweisen, z.B.:

```
<textarea name="FeldA1" rows="20" cols="100" class="fullwidth">
</textarea>
```

Diese Einstellung wirkt bereits für Plaintext-Textareas, wird aber auch beim Laden von WYSIWYG-Texteditoren berücksichtigt: Die Boxen passen sich immer an die Seitenbreite (bzw. die Absatzbreite) an.

Beschränken des Editors auf bestimmte Eingabeboxen

Falls nicht *alle* Textareas der Seite mit einem WYSIWYG-Editor ausgestattet werden sollen, markieren Sie jede mit WYSIWYG-Editor auszustattende Textarea, indem Sie das Attribut `class="WYSIWYG"` zum Textarea-Tag hinzufügen, und geben Sie im oben beschriebenen Kommentar zusätzlich das Wort `class` mit an. Alternativ können Sie auch andere Klassennamen verwenden, indem Sie `class="irgendwas"` sowohl zum WYSIWYG-Kommentar als auch zu jeder mit dem Editor auszustattenden Textarea hinzufügen.

Wichtig: Falls Ihr Editor die Anpassung an die Seitenbreite verwendet, Sie also im Class-Attribut des

`<textarea…>`-Tags neben der Klasse `WYSIWYG` auch die Klasse `fullwidth` angegeben haben, kopieren Sie das gesamte Class-Attribut in den WYSIWYG-Kommentar, z.B. :

```
<!-- WYSIWYG … class="WYSIWYG fullwidth" -->
…
<textarea … class="WYSIWYG fullwidth">
</textarea>
```

Der Klassenname `WYSIWYG` ist dabei auch gegen beliebige andere Namen austauschbar.

(Die alleinige Angabe von `class` ohne Wertzuweisung im Kommentar ist gleichbedeutend mit `class="WYSIWYG"` ohne Angabe von `fullwidth`.)

Laden verschieden konfigurierter Editoren für bestimmte Eingabeboxen

Falls Sie unterschiedliche Editor-Konfigurationen für verschiedene Textareas einer Seite laden möchten, fügen Sie mehrere WYSIWYG-Kommentare hintereinander in die Seite ein, einen für jede zu verwendende Editor-Konfiguration. Geben Sie in jedem der Kommentare den Zusatz `class="`*Name*`"` an, wobei Sie für jeden Kommentar einen anderen Namen in die Anführungszeichen einsetzen. Dieser Name identifiziert Ihre jeweilige Editor-Konfiguration. Fügen Sie nun zum öffnenden Tag jeder Textarea, die mit einem WYSIWYG-Editor ausgestattet werden sollen, dasselbe class-Attribut hinzu wie in dem jeweiligen WYSIWYG-Kommentar.

Ein Beispiel:

```
<html>
 <head>
  …
  <!-- WYSIWYG class="einfacherEditor" -->
  <!-- WYSIWYG erweitert absatzformate autogroesse class="erweiterterEditor fullwidth"--
>
 </head>
 <body>
  …
  <textarea class="einfacherEditor">Diese Box nutzt den einfachen Editor</textarea>
  <textarea class="erweiterterEditor fullwidth">Diese Box nutzt den erweiterten Editor
mit Absatzformatauswahl.</textarea>
  <textarea>Diese Box nutzt gar keinen WYSIWYG-Editor.</textarea>
  …
 </body>
</html>
```

Erzwingen eines älteren Editors

Seit Januar 2023 wird standardmäßig TinyMCE 5 als Editor eingesetzt. Zwischen April 2014 und Januar 2023 kam TinyMCE 4 und vorher TinyMCE 3 zum Einsatz.

Wenn Sie den Editor wie oben beschrieben einbinden, wird in Ihren Seite immer der gerade aktuelle Editor benutzt.

Falls Sie aber schon vor einer dieser Umstellungen den Editor benutzt hatten und eine automatische Umstellung auf den neueren aus irgendwelchen Gründen verhindern oder rückgängig machen möchten, können Sie in Ihren Seiten die Nutzung des alten TinyMCE-3- oder TinyMCE-4-Editors erzwingen. Fügen Sie dazu – zusätzlich zu den oben beschriebenen Kommentaren zur Editor-Konfiguration – einen weiteren Kommentar in den Seitenkopf ein:

```
<!-- WYSIWYG-EDITOR tinymce4 -->
```

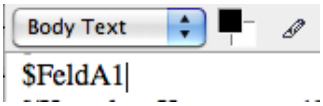
bzw.

```
<!-- WYSIWYG-EDITOR tinymce3 -->
```

Anpassung von Quittungs- und Korrekturseiten

Falls in einer Aufgabenseite ein HTML-Editor für studentische Eingaben eingebunden wurde, sind auch noch die Quittungs- und Korrekturseiten entsprechend anzupassen, so dass die HTML-Eingaben der Studenten auch korrekt formatiert dargestellt werden. Zu diesem Zweck sind folgende Punkte zu beachten:

- Die `$Feld`-Variablen zu den mit HTML-Editor ausgestatteten Feldern (also allen Textareas) *dürfen nicht auf ‚P' enden!*
  Das Suffix ‚P' ist ideal für Plaintext. Es bewirkt, dass alle in der Einsendung auftretenden ‚<'- und ‚>'-Zeichen durch `<` bzw. `>` ersetzt werden, also in der Korrektur als Zeichen dargestellt und nicht vom Browser als Tag-Begrenzungen interpretiert werden. Bei HTML-Eingaben eines Studenten dagegen ist das P-Suffix unerwünscht, denn dann würde dem Korrektor der HTML-Quelltext der studentischen Einsendung angezeigt, der Browser würde die HTML-Tags, die der Editor eingefügt hat, nicht interpretieren.
- Die `$Feld`-Variablen *dürfen auch nicht*, wie für Plaintext-Eingaben normalerweise voreingestellt, *als „preformatted text" formatiert sein* (also innerhalb eines `<pre>`-Elements stehen)! Das pre-Element verhindert insb. ein Umwandeln der Eingaben durch den Browser in Fließtext, erhält dafür aber die Zeilenumbrüche in Plain-Text-Eingaben – was bei HTML-Eingaben nicht nötig ist.
- *Ebenso wenig darf eine solche* `$Feld`*-Variable als Absatz* (`<p>`) *oder Überschrift formatiert sein*, da sie durch eine Folge von Absätzen (die studentische Eingabe) ersetzt wird.
- *Die Variable sollte daher vielmehr als „Body Text" formatiert sein*, d.h. direkt innerhalb des `<body>`-Elements stehen oder innerhalb von Blockelementen, die ihrerseits Absätze, Überschriften etc. enthalten dürfen (z.B. `<div>` oder `<li>`).
  Falls Sie die Aufgabenseite mit dem SeaMonkey Composer im WYSIWYG-Modus bearbeiten, achten Sie darauf, dass für die Variable in der Auswahlbox oben links das Absatzformat „Body Text" eingestellt ist, wie in nebenstehender Abbildung gezeigt.



In-Browser-Korrektur

Wie bereits in der Einleitung beschrieben, ist die In-Browser-Korrektur ein alternatives Korrekturverfahren, bei dem eine Korrekturkraft direkt im Browser korrigiert. Sie als Aufgabenautor müssen dazu zunächst in den Korrekturschablonen Eingabemöglichkeiten für Korrekturkräfte vorsehen. Ein Korrektor bekommt dann bei Auswahl einer zu korrigierenden Einsendung nicht mehr die „nackte" Korrekturseite zum Laden und Bearbeiten in einem externen HTML-Editor angezeigt, sondern die ihm angezeigte Korrekturseite enthält die von Ihnen vorgesehenen Eingabefelder sowie einen Button zum Speichern der Korrektur. Nach dem Speichern erhält der Korrektor eine Vorschau auf die Korrektur, wie der Student sie nach Freigabe sehen wird: An Stelle der Eingabefelder werden hier die Korrektoreingaben direkt (und automatisch hervorgehoben) angezeigt.

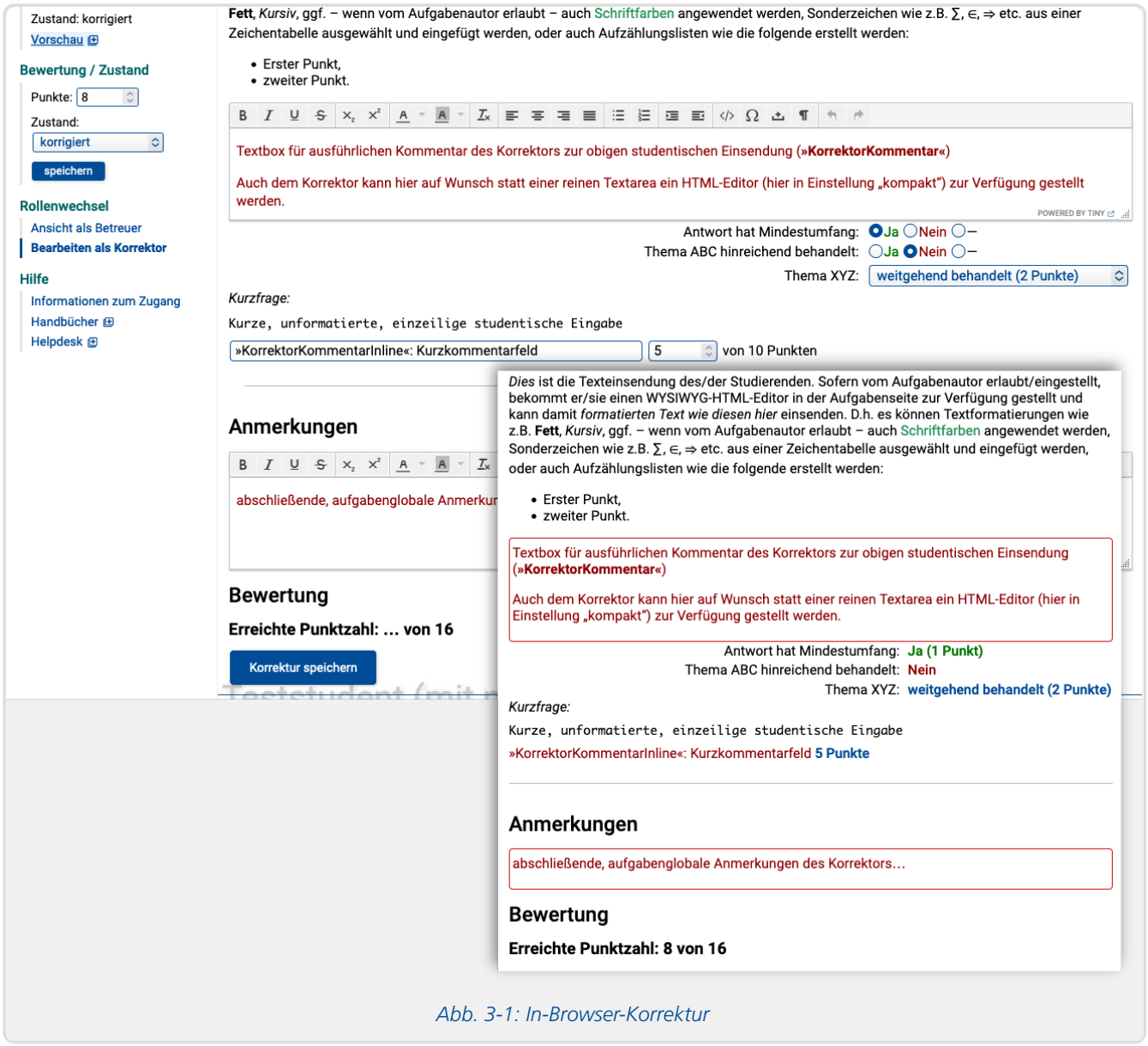


Abb. 3-1: In-Browser-Korrektur

Obige Abbildung zeigt exemplarisch die Ansicht des Korrektors / der Korrektorin (mit Eingabeboxen) und die Ansicht der fertigen Korrektur, wie der/die Student/-in sie sehen wird.

Die In-Browser-Korrektor bietet Ihnen verschiedene Arten von Eingabemöglichkeiten für Korrektoren an (i.F. als *„In-Browser-Korrektur-Elemente"* bezeichnet), die Sie in der Korrekturschablone vorsehen können:

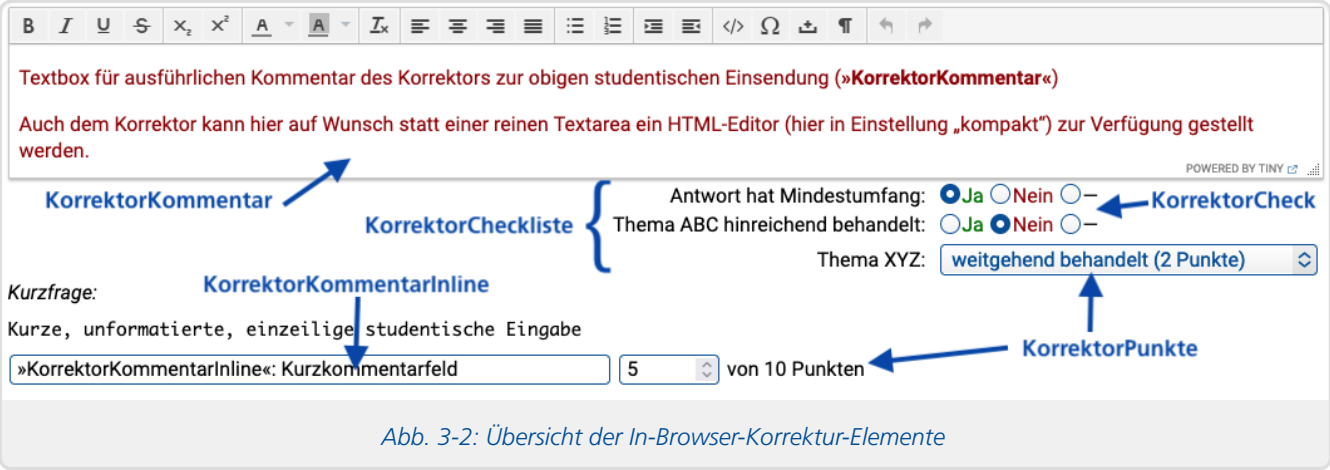


Abb. 3-2: Übersicht der In-Browser-Korrektur-Elemente

1. Ein `KorrektorKommentar` dient zur Aufnahme eines aus einem oder mehreren Absätzen bestehenden Textes. In der Korrektoransicht wird hierzu eine Textbox dargestellt, die wahlweise mit einem der im vorigen Kapitel beschriebenen HTML-WYSIWYG-Editoren ausgestattet werden kann. In der Studentenansicht wird der Kommentar als optisch hervorgehobener Block dargestellt. (Die Standardschriftfarbe ist rot, um in der Korrekturansicht die vom Korrektor/der Korrektorin eingegebenen Texte besser von den vom Student/der Studentin eingesendeten Texte unterscheiden zu können. Prinzipiell haben Sie aber die Möglichkeit, in Ihrem Kurs eine abweichende Formatvorlage im CSS-Format zu installieren.)
2. Ein `KorrektorKommentarInline` dient zur Aufnahme eines kurzen Klartext-Kommentars, der insbesondere nicht aus Absätzen besteht und in der Korrektur innerhalb einer bestehenden Zeile eingefügt werden kann, also kein HTML-Block-Element bildet. In der Korrektoransicht wird hierzu eine einfache Texteingabezeile dargestellt. In der Studentenansicht wird der Kommentar als Inline-Element in den umgebenden Text eingefügt.
3. Ein `KorrektorCheck` dient zur Aufnahme einer „Richtig-oder-Falsch-Kennzeichnung“. Dem Korrektor werden hierzu zwei Radiobuttons »Richtig« und »Falsch« angezeigt, und er kann nach Durchsicht des Teils der Einsendung, auf die sich dieser KorrektorCheck bezieht, markieren, ob die Lösung korrekt ist oder nicht. So lange er keinen von beiden Radiobuttons markiert hat, gilt die Antwort als unkorrigiert. (Es gibt noch einen dritten, mit »–« beschrifteten Radiobutton, der lediglich dazu dient, eine bereits eingegebene Richtig-/Falscheingabe wieder zu entfernen.)
    - Die Beschriftungen „Richtig“ bzw. „Falsch“ der Radiobuttons können Sie als Aufgabenautor beliebig austauschen.
    - Außerdem können Sie einem KorrektorCheck Punkte zuordnen.
    - Dem/-r Studenten/-in wird an der Stelle des KorrektorChecks im Normalfall je nach Markierung des/-r Korrektors/-in das Wort „Richtig“ oder „Falsch“ angezeigt, bzw. der jeweilige vom Aufgabenautor definierte Alternativtext. Sind dem KorrektorCheck Punkte zugeordnet, wird bei richtiger Antwort in der Regel auch die mit dieser richtigen Antwort erreichte Punktzahl hinter dem Wort „Richtig“ angezeigt, z.B.: „Richtig (5 Punkte)“.
    - Sowohl die Anzeige der Punkte als auch die Anzeige von „Richtig“ oder „Falsch“ lässt sich auch abschalten. So ist es insbesondere möglich, die KorrektorChecks nur dem Korrektor anzuzeigen, um die Punktzahl zu berechnen, ohne dass der Student diese zu sehen bekommt.
4. Ein `KorrektorPunkte`-Element kann in zwei Varianten vorkommen:
    - Eingabefeld, in das eine Punktzahl eingetippt werden kann, wahlweise durch eine einstellbare Maximalpunktzahl begrenzt,
    - Auswahlbox, in der ein/-e Korrektor/-in zwischen verschiedenen Bewertungen auswählen kann. In der Regel sind diese Bewertungen durch einen Text „beschriftet“, und ihnen ist jeweils eine zu vergebende Punktzahl zugeordnet (die wahlweise dem/der Studenten/-in mit angezeigt oder verborgen werden kann).
      Diese Form ist gewissermaßen vergleichbar mit `KorrektorCheck`-Elementen, nur eben mit mehr Auswahlmöglichkeiten (neben falsch und richtig), und ausschließlich mit Punktevergabe kombinierbar.
5. Eine Folge von mehreren `KorrektorPunkte`- oder `KorrektorCheck`-Elementen – auch gemischt –

kann zu einem `KorrektorCheckliste`-Block zusammengefasst werden. Das bewirkt im Wesentlichen, dass diese bündig untereinander ausgerichtet werden (siehe Abbildungen 3-1 und 3-2 oben).

`KorrektorPunkte`-Elemente werden nicht unbedingt benötigt: Wenn Sie auf sie verzichten, hat der Korrektor immer noch die Möglichkeit, eine Gesamtpunktzahl zur korrigierten Aufgabeneinsendung zu vergeben.
Aber wenn Ihre Aufgabe sich z.B. in mehrere Fragen/Teilaufgaben gliedert, können Sie über `KorrektorPunkte` den Korrektoren/-innen die Möglichkeit geben, die Punktzahl zu jeder Teilaufgabe einzeln zu vergeben. Die Gesamtpunktzahl wird dann automatisch als Summe der Teilpunkte berechnet.

Eine andere Anwendung ist z.B., falls Sie die Wertung zu einer Einsendung in mehrere Teilbewertungen nach bestimmten Bewertungskriterien (wie Stil, Umfang, Verständlichkeit, Quellenbezüge etc. von Textausarbeitungen) aufgliedern möchten. Dann können Sie eine Reihe von `KorrektorPunkte`- oder (bei einfachen Ja-/Nein-Kriterien) `KorrektorCheck`-Elementen untereinander als `KorrektorCheckliste` zusammenstellen, wie in der obigen Abbildung anhand dreier Beispielkriterien demonstriert.

Falls Sie zu einer Aufgabenseite `KorrektorPunkte`- oder `KorrektorCheck`-Elemente mit zugeordneter Punktzahl bei richtiger Antwort hinzugefügt haben, so gilt:
Wird eine Korrektur gespeichert, so wird die Summe der Punkte aller in der Korrektur als richtig markierten `KorrektorCheck`s und aller Punkteeingaben oder -Auswahlen in `KorrektorPunkte`n automatisch als erreichte Gesamtpunktzahl für die Korrektur eingetragen. Auch wird dann beim Speichern einer Korrektur ihr Status ggf. automatisch auf „vorläufig korrigiert“ oder „korrigiert“ gesetzt. Der Zustand „korrigiert“ wird dabei vergeben, wenn vom Korrektor *alle* KorrektorCheck- und KorrektorPunkte-Felder ausgefüllt und dabei eine Punktzahl berechnet wurde. Falls der/die Korrektor/-in noch Felder unausgefüllt gelassen hat oder wenn gar keine KorrektorPunkte und höchstens KorrektorChecks ohne `punkte`-Attribut in der Korrekturseite vorkommen, wird der Korrekturstatus lediglich von „unkorrigiert“ auf „vorläufig korrigiert“ angehoben, aber nicht auf „korrigiert“ gesetzt, da davon ausgegangen wird, dass die Bewertung damit noch nicht endgültig ist (noch Teilpunkte fehlen oder eben gar keine Teilpunkte-Eingabemöglichkeiten bestehen und somit der Korrektor die Gesamtpunktzahl noch separat erfassen muss).

In den folgenden Abschnitten wird die Einbindung/Konfiguration der In-Browser-Korrektur-Elemente bei der Aufgabenerstellung beschrieben.

Assistentengestützte Einbindung

Im Aufgabenerstellungsassistenten des Online-Übungssystems für handbewertete Aufgaben können Sie am Fuße der Seite unter „Korrekturmodus“ die In-Browser-Korrektur aktivieren. Damit wird bereits eine KorrektorKommentar-Box am Ende der Korrekturseite eingefügt (vgl. Abbildung oben unter Überschrift »Anmerkungen«). Weitere Elemente können Sie einzeln zur Aufgabenseite hinzufügen:

- Das Element »*Korrektorkommentar-Eingabebox*« erzeugt eine hier als `KorrektorKommentar` beschriebene Textarea,
- Das Element »*KorrektorCheck*« erzeugt ein `KorrektorCheck`.
- Das Element »*Teilpunkte-Eingabe bzw. -Auswahl*« erzeugt ein `KorrektorPunkte`-Element.
- `KorrektorKommentarInline`-Elemente können Sie nicht einzeln hinzufügen, sondern im Assistenten lediglich einer einzeiligen Eingabemöglichkeit für Studenten anfügen, indem Sie im Element »*Eingabefeld (einzeilig)*« die Zusatzoption „Korrektor-Kurzkommentar ermöglichen“ markieren.
- Falls Sie direkt hinter einem »*Eingabefeld (einzeilig)*« (für Studenten-Eingaben) ein »KorrektorCheck«- oder »Teilpunkte-…«- Element einfügen, bekommen Sie dort jeweils die Option angezeigt, dass dieses (in der Korrekturseite) in keiner eigenen Zeile stehen, sondern hinter die Eingabe aus dem Studenten-Eingabefeld gesetzt werden soll. (Das ist in der obigen Beispielabbildung demonstriert, in der sich ein `KorrektorKommentarInline` und ein `KorrektorPunkte`-Element hinter einer studentischen einzigen Kurzeingabe in einer Zeile befinden. Es können mehrere solcher Zeilen aus Kurzeingaben, Kurzkommentar und/oder KorrektorCheck/-Punkte aufeinander folgen.)
- Ein `KorrektorCheckliste`-Block wird vom Aufgabenerstellungsassistenten automatisch erzeugt, wenn Sie mehrere »KorrektorCheck«- oder »Teilpunkte-…«Elemente hintereinander in die Aufgabenseite

einfügen. In so einer Checkliste ist es sinnvoll, jeweils das Feld »Voranzustellender Text« auszufüllen, als Beschriftung für das jeweilige Bewertungskriterium.

Die beiden folgenden Screenshots zeigt die typsichen Elemente zur In-Browser-Korrektur im Aufgabenerstellungs-Assistenten (als Ausschnitte der Aufgabenkonfiguration zu den obigen Beispielabbildungen der Korrektoren- und Studentensicht):

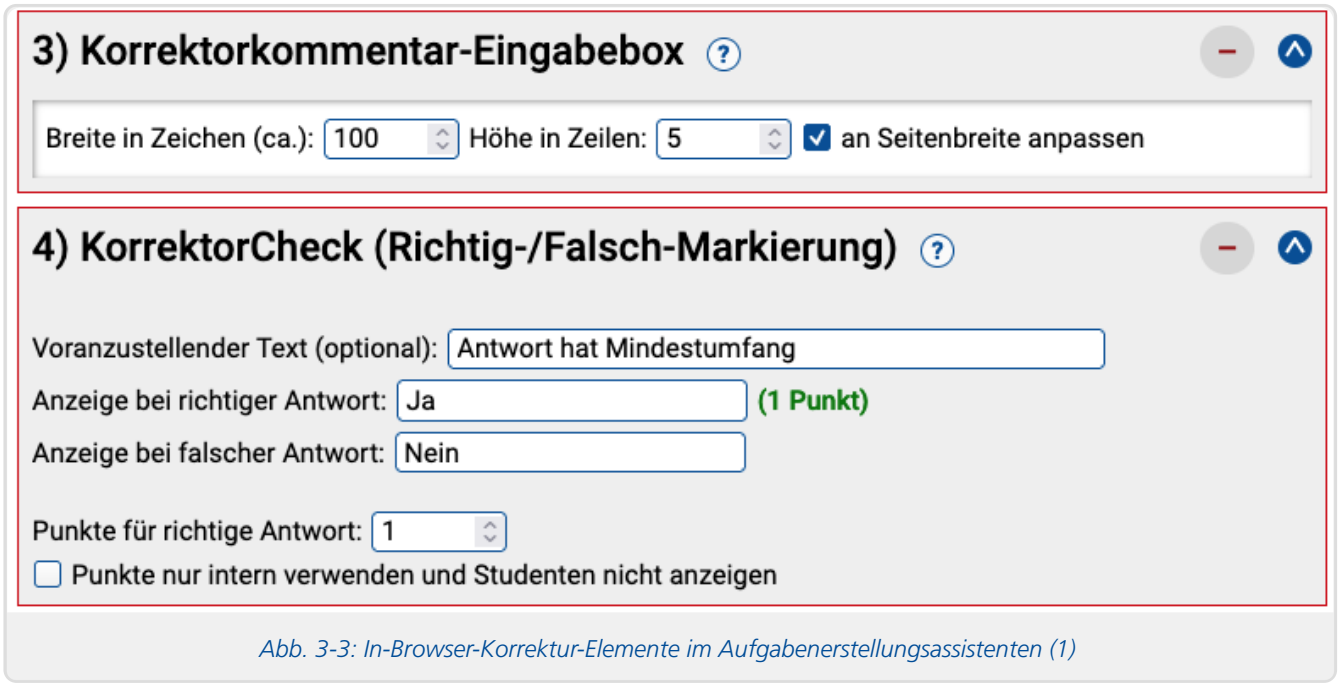


Abb. 3-3: In-Browser-Korrektur-Elemente im Aufgabenerstellungsassistenten (1)

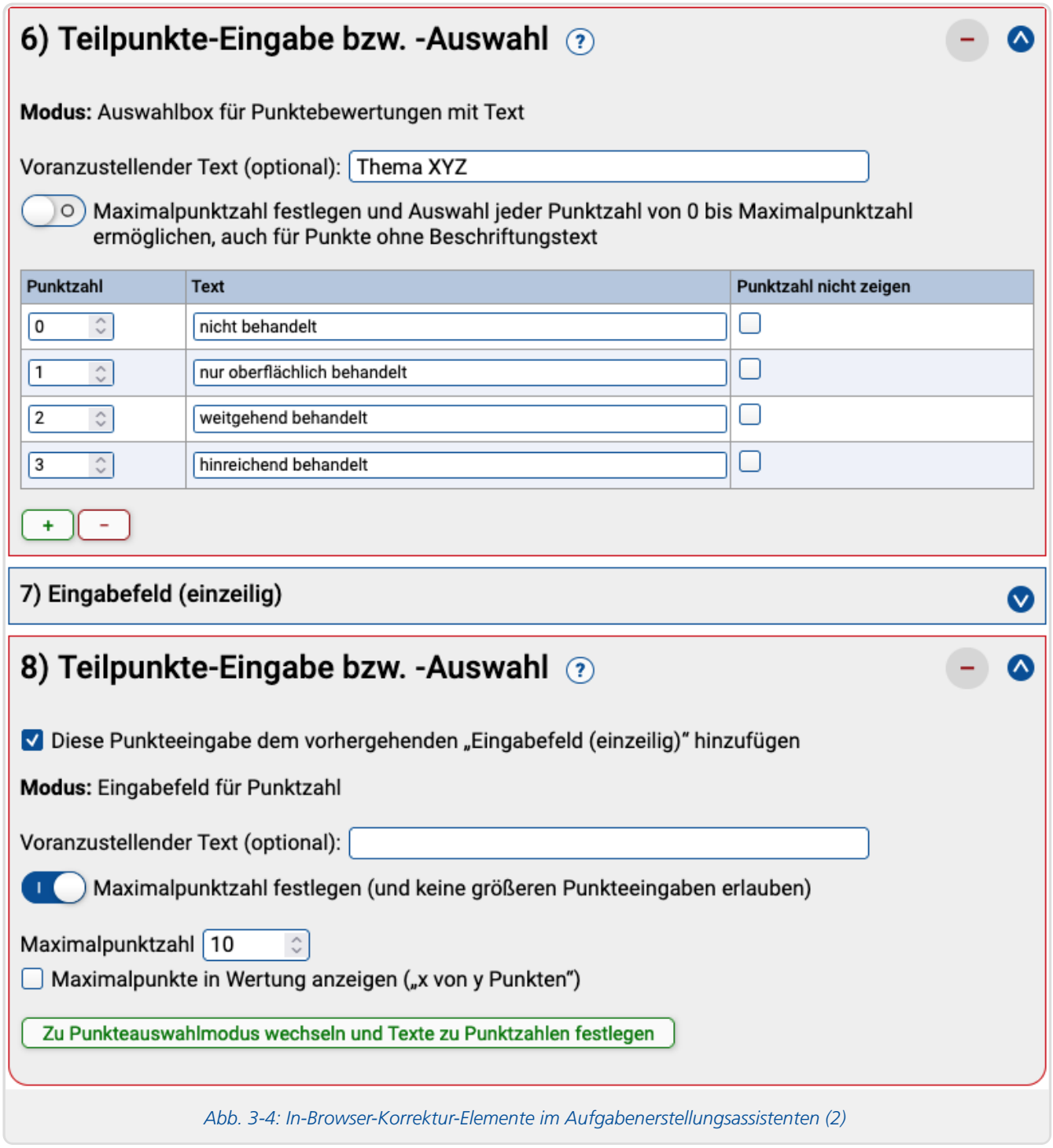


Abb. 3-4: In-Browser-Korrektur-Elemente im Aufgabenerstellungsassistenten (2)

Falls Ihre Aufgabenseite aus mehreren Fragen (eingeleitet durch separate »Überschrift: Neue Frage beginnen«-Elemente) besteht und die Korrektoren einfach zu jeder Frage genau eine Teilpunktzahl (Fragenpunktzahl) einzeln erfassen können sollen, so müssen Sie die KorrektorPunkte-Elemente (»Teilpunkte-Eingabe oder -Auswahl«) nicht für jede Frage separat von Hand hinzufügen. Für diesen Fall gibt es unten in der Editorseite im Abschnitt »Korrekturmodus« als Shortcut einen Button, der einfach jede Frage der Aufgabenseite um ein Punkteeingabefeld ergänzt:

Abb. 3-5: KorrektorPunkte zu jeder Frage der Seite hinzufpgen

Dieser Button steht nur zur Verfügung, wenn die In-Browser-Korrektur aktiviert ist und die Seite aus mehr als einer Frage besteht.

Manuelle Einbindung in HTML-Code

KorrektorKommentar einbinden

Syntax

Um einen KorrektorKommentar in Ihre Korrekturschablone einzubinden, notieren Sie dort ein Element der folgenden Art:

- `[KorrektorKommentar`# *Attribute*`/]` oder
- `[KorrektorKommentar`# *Attribute*`]` *Inhalt* `[/KorrektorKommentar`#`]`

Dabei ist jeweils # durch eine Nummer zu ersetzen, die den KorrektorKommentar eindeutig identifiziert (also nicht bereits für einen anderen KorrektorKommentar verwendet wurde).

An der Stelle *Attribute* können Sie optional Attribute für das HTML-Textarea-Tag angeben: Alle Attribute, die Sie hier einfügen, werden in der Korrektoransicht 1:1 ins HTML-Tag übernommen. Insbesondere können Sie durch Angabe der Attribute `cols` und `rows` die Größe für die Textarea festlegen. Geben Sie diese Attribute nicht an, bekommt die Textarea eine vom Online-Übungssystem (nicht vom Browser) festgesetzte Defaultgröße. Neben `cols` und `rows` sind alle Attribute erlaubt, die fürs Textarea-Tag erlaubt sind, mit Ausnahme des Attributs name, das vom Online-Übungssystem vergeben wird!

Hinweis: Wenn Sie den KorrektorKommentar mit einem WYSIWYG-Editor kombinieren, wird (zumindest seit TinyMCE Version 4) der Editor sich in der Breite immer an die Seitenbreite anpassen, eine `cols`-Angabe also ignoriert. Aber für Plaintext-Eingabefelder ist die Angabe ggf. sinnvoll.

Falls Sie die Textbox für den Korrektor bereits mit einem*Inhalt* vorfüllen möchten, den dieser später ersetzen bzw. bearbeiten kann, dann geben Sie diesen wie oben gezeigt zwischen einem öffnenden und einem schließenden KorrektorKommentar#-Tag an. Andernfalls ist die Kurzschreibweise mit atomarem Element möglich.

HTML-Kontext

Ein `KorrektorKommentar`-Element wird in der Studentenansicht durch ein `div`-Element ersetzt. Es darf daher nur an Stellen im HTML-Code der Schablone notiert werden, an denen nach HTML-Standard auch `div`-Elemente stehen dürfen.

Typischerweise wird ein KorrektorKommentar direkt als „Body Text“ notiert, also als direktes Kind des Body-Elements. Insbesondere nicht erlaubt ist die Positionierung innerhalb eines Absatzes oder z.B. eines pre-Elements für vorformatierten Text! (Falls Sie die Korrekturschablone mit dem SeaMonkey Composer im WYSIWYG-Modus bearbeiten, achten Sie darauf, dass in der Auswahlbox oben links das Absatzformat „Body Text“ eingestellt ist (vgl. obige Abbildung).

Sollen alle KorrektorKommentare in der Korrekturseite mit einem WYSIWYG-Editor ausgestattet werden, binden Sie außerdem in der Korrekturseite noch den entsprechenden Kommentar ein (vgl. obiges Kapitel zu WYSIWYG-Editoren).

Beispiele

Der folgende Beispiel-Ausschnitt zeigt die Einbindung eines Korrektorkommentars mit expliziter Angabe der Textbox-Größe (10 Zeilen à 100 Zeichen) und vorgefüllt mit Text für einen abschließenden Korrektor-Kommentar vor der Angabe der erreichten Punkte am Ende der Korrekturseite:

```
<h3>Anmerkungen</h3>
[KorrektorKommentar3 cols="100" rows="10"]{hier bitte zus&auml;tzl. Anmerkungen
eintragen}[/KorrektorKommentar3]
<h3>Erreichte Punktzahl: $KorrekturPunkte von $MaximalPunkte</h3>
```

Dasselbe Beispiel noch einmal, nur ohne explizite Größenangabe und ohne vorgefüllten Inhalt:

```
<h3>Anmerkungen</h3>
[KorrektorKommentar3 /]
<h3>Erreichte Punktzahl: $KorrekturPunkte von $MaximalPunkte</h3>
```

KorrektorKommentarInline einbinden

Syntax

Die Syntax ist praktisch identisch zu der Einbindung eines KorrektorKommentars:

- `[KorrektorKommentarInline`# *Attribute* `/]` oder
- `[KorrektorKommentarInline`# *Attribute* `]` *Inhalt* `[/KorrektorKommentarInline`# `]`

Dabei ist wieder jeweils # durch eine Nummer zu ersetzen, die den KorrektorKommentarInline eindeutig identifiziert.

Attribute werden 1:1 ins HTML-Input-Tag übernommen, d.h. hier dürfen alle Attribute angegeben werden, die in HTML-Input-Tags erlaubt sind (mit Ausnahme der Attribute `type`, `name` und `value`). Insbesondere können Sie per size die Größe der Eingabezeile festlegen oder auch per maxlength (maximale Länge der Eingabe in Zeichen) dem Korrektor ein Limit für die Kommentarlänge setzen.

HTML-Kontext

Ein `KorrektorKommentarInline`-Element wird in der Studentenansicht durch ein Inline-Element (`span`) ersetzt. Es darf – anders als ein `KorrektorKommentar` – auch innerhalb von Absätzen oder vorformatiertem Text, Tabellen etc. vorkommen, überall, wo auch span-Elemente zulässig sind.

Beispiel

```
<p>Kommentar des Korrektors: [KorrektorKommentarInline1 size="70" /]</p>
```

KorrektorCheck einbinden

Syntax

Zur Einbindung eines `KorrektorChecks` in die Korrekturschablone notieren Sie ein Element mit folgendem Aufbau:

- `[KorrektorCheck`# *Attribute* `/]`

Dabei ist # wieder durch eine Nummer zu ersetzen, welche das KorrektorCheck-Element eindeutig identifiziert. Der schließende Slash (/) ist beim `KorrektorCheck`-Element optional (darf also entfallen) – anders als bei den `KorrektorKommentar(Inline)`-Elementen, bei denen, falls kein Slash vor der schließenden Klammer des öffnenden Tags steht, ein entsprechendes schließendes Tag am Ende des Vorgabetextes erwartet wird.

Die folgenden optionalen *Attribute* sind möglich:

Attribut	Bedeutung
`richtig`	Alternative Beschriftung an Stelle des Wortes „Richtig“. Kann leer sein (`richtig = ""`). Falls das Attribut auf Minus (–) endet, wird die Ausgabe der Punktzahl unterdrückt (s.u.). Das Minus selbst wird nicht mit ausgegeben.
`falsch`	Alternative Beschriftung an Stelle des Wortes „Falsch“. Kann leer sein (`falsch = ""`).
`punkte`	Zu vergebende Punktzahl bei richtiger Antwort. Ordnen Sie damit Ihren KorrektorChecks Punkte zu, falls die Gesamtpunktzahl der Aufgabe sich als Summe der Teilpunkte für alle richtigen Antworten errechnen soll.
`praefix`	Ein Text, der den Radiobuttons in der Korrektorensicht sowie auch der Anzeige in der Studentensicht (sofern diese nicht leer / unsichtbar ist) vorangestellt wird. Insbesondere in KorrektorChecklisten sollten Präfixe verwendet werden, um die einzelnen KorrektorChecks dieser Liste (z.B. mit verschiedenen Bewertungskriterien) zu beschriften. Für *einzelne* KorrektorChecks hinter einer studentischen Eingabe wird normalerweise kein Präfix benötigt, da der Bezug i.d.R. klar ist.
`prefix`	Synonym zu `praefix`, beide Schreibweisen sind gleichwertig.
`block`	Flag-Attribut ohne Wertzuweisung. Wenn angegeben, wird das KorrektorCheck-Element als Blockelement (div) gerendert (und per Default-CSS am rechten Seitenrand ausgerichtet), ohne Angabe wird es als Inlineelement (span) eingefügt. Ausnahme: Innerhalb einer KorrektorCheckliste hat dieses Attribut keine Auswirkung.
`value`	Zulässige Werte (unabhängig von der mit `richtig` oder `falsch` festgelegten Beschriftung): „Richtig“ oder „Falsch“. Hiermit wird ein (Vorgabe-)Wert festgelegt. Im Normalfall ist von der Angabe dieses Attributs in Korrekturschablonen abzuraten, denn nur, wenn alle KorrektorChecks zu Beginn unausgefüllt sind (also kein `value`-Attribut besitzen), kann der Korrektor erkennen, welche er schon bearbeitet (ausgefüllt) hat und welche nicht.

Der folgende Unterabschnitt geht noch genauer auf den Einsatz der ersten drei Attribute ein:

Beschriftungen der KorrektorCheck-Radiobuttons

Die Standardbeschriftungen lauten „Richtig“ bzw. „Falsch“, können aber durch Angabe der oben genannten Attribute `richtig` bzw. `falsch` angepasst werden. Die Beschriftung „Super!“ bei richtiger Antwort z.B. legt man demnach durch Angabe des Attributs `richtig="Super!"` fest. Im Normalfall stimmen die Beschriftungen der Radiobuttons in der Korrektorsicht und die je nach markiertem Radiobutton dem Studenten angezeigten Werte überein, d.h. wenn der Korrektor den Radiobutton mit der Beschriftung „Super!“ selektiert, wird auch dem Studenten der Text „Super!“ in seiner Korrektur angezeigt. Eine Ausnahme liegt vor, falls Sie „leere Beschriftungen“ festlegen (z.B. `falsch=""`). Dann wird dem Studenten kein Text angezeigt, die entsprechende Checkbox für den Korrektor trägt dann wieder die Default-Beschriftung („Richtig“ bzw. „Falsch“).

Falls das Attribut `punkte` angegeben und somit eine erreichbare Punktzahl festgelegt wurde, wird in der dem

Studenten angezeigten Korrektur im Falle einer richtigen Antwort hinter dem Wort „Richtig“ bzw. der festgelegten Alternativbeschriftung auch noch in Klammern die Punktzahl angezeigt, z.B.: „Richtig (5 Punkte)“. Bei Angabe von `richtig=""` und Angabe des `punkte`-Attributs wird dem Studenten bei richtiger Antwort nur die Punktzahl in Klammern angezeigt, z.B. „(5 Punkte)“.

Sollen die den KorrektorChecks zugeordneten Punkte zwar intern zur Berechnung des Gesamtergebnisses der Aufgabe herangezogen, für die Studenten jedoch nicht sichtbar sein, so legen Sie per Attribut `richtig` eine Beschriftung für richtige Antworten fest und hängen Sie dieser Beschriftung ein ‚-‘ an, z.B. `richtig="Richtig-"` (zur Ausgabe nur des Wortes „Richtig“ ohne Punktzahl) oder `richtig="-"` (falls gar nichts angezeigt werden soll, weder eine Beschriftung noch die erreichte Punktzahl). (Bei falschen Antworten entfällt diese Sonderregelung mit dem ‚-‘-Suffix, da dort ohnehin niemals eine Punktzahl angezeigt wird.)

Punktzahlen mit Nachkommastellen

Seit Herbst 2022 unterstützt das Online-Übungssystem prinzipiell, dass Korrekturpunkte auch Nachkommastellen haben dürfen. Die insgesamt erreichbaren Maximalpunkte zu einer Aufgabe müssen nach wie vor ganzzahlig sein, aber die vom Korrektor (oder auch einem Korrekturmodul bei Autokorrektur) vergebenen tatsächlich erreichten Punkte (zwischen 0 und der erreichbaren Maximalpunktzahl) müssen ggf. nicht mehr unbedingt ganzzahlig sein.
Tatsächlich sind ganzzahlige Korrekturpunkte nach wie vor die Grundeinstellung einer Kursumgebung, aber in den Kursparametern kann auch eingestellt werden, dass eine oder zwei Nachkommastellen erlaubt sein sollen.

Wenn Ihr Kurs also ein Punkteformat mit Nachkommastelle(n) verwendet, dann dürfen auch die mit einem KorrektorCheck zu vergebenden Punkte entsprechend 1 oder 2 Nachkommastellen haben. Geben Sie in diesem Fall im `punkte`-Attribut eine Zahl mit Komma als Dezimaltrenner ein, z.B. `punkte="0,5"`, wenn die richtige Markierung einen halben Punkt wert sein soll.

Dabei sind insbesondere folgende Besonderheiten zu beachten:

1. Falls die Korrektoren/-innen selbst keine Punktzahlen eingeben können, sondern lediglich KorrektorChecks markieren oder aus KorrektorPunkte-Selectboxen auswählen können, und Sie hier (oder bei den KorrektorPunkte-Elementen) keine ganzen Punktzahlen verwenden, könnte es vorkommen, dass die beste insgesamt erreichbare Punktzahl nicht ganzzahlig wäre. Beispiel: Müssen die Korrektoren/-innen 5 KorrektorChecks ausfüllen, die jeweils z.B. 2,5 Punkte vergeben, wären bestenfalls 12,5 Punkte insgesamt erreichbar. Allerdings muss die in einer Aufgabe insgesamt maximal erreichbare Punktzahl *immer ganzzahlig* sein! Achten Sie bei solchen Konstellationen daher darauf, dass sich die erreichbaren nicht-ganzzahligen Teilpunkte insgesamt immer zu einer ganzen Maximalpunktzahl summieren! Andernfalls könnten die Studierenden nie die als ganze Zahl festgelegte Maximalpunktzahl erreichen.
2. *Falls Sie hier im HTML Punktzahlen mit mehr Nachkommastellen eintragen als laut den Kursparametern für Korrekturpunkte erlaubt sind*, rundet das Übungssystem automatisch auf die zulässige Nachkommastellenzahl. Im Falle von `punkte="7,25"` z.B. werden bei Erlaubnis für zwei Nachkommastellen tatsächlich 7,25 Punkte vergeben, bei Einstellung auf nur *eine* Nachkommstelle würden bei „Richtig“-Markierung nur 7,3 Punkte vergeben, und wenn gar keine Nachkommastellen erlaubt sind, dann 7 Punkte.
   Natürlich sollten solche Konstellationen normalerweise vermieden werden, können aber durchaus vorkommen, wenn Sie z.B. in den Kursparametern die erlaubte Anzahl von Nachkommastellen nachträglich reduzieren oder Aufgaben/Fragen in einen anderen Kurs mit anderer eingestellter Nachkommastellenzahl importieren.

HTML-Kontext

Per Default wird ein `KorrektorCheck` (analog zu einem `KorrektorKommentarInline`) als Inline-Element (`span`) in den HTML-Code eingebunden und darf daher überall vorkommen, wo span-Elemente erlaubt sind, insbesondere innerhalb von Absätzen oder auch Überschriften.

Alternativ können Sie ein `KorrektorCheck` aber auch als Blockelement (`div`) einbinden lassen, indem Sie

im Tag das Attribut `block` mit aufnehmen. Dieses Blockelement wird mit einer Klasse versehen und per Default rechtsbündig in der Korrekturseite ausgerichtet. (Die rechtsbündige Ausrichtung ist daran angelehnt, dass auch bei handschriftlichen Korrekturen kurze Korrekturvermerke, Häkchen oder Punktwertungen meist an den rechten Rand geschrieben werden. Siehe CSS-Anpassung, falls Sie davon abweichen möchten.)

Sollten Sie das `KorrektorCheck` innerhalb einer `KorrektorCheckliste` einbauen, spielt das `block`-Attribut keine Rolle (wird ignoriert), das KorrektorCheck wird dann immer als Block mit zwei Inline-Abschnitten für Präfix und Radiobuttons bzw. Anzeige gerendert, die eine tabellarische Darstellung ermöglichen.

Beispiele

Die einfachste Variante ist ein KorrektorCheck ohne Attribute, das folgende Beispiel zeigt dessen Einbindung in die Überschrift zu einer Teilaufgabe:

```
<h4>Teil a) [KorrektorCheck1 /]</h4><pre>$FeldA1P</pre>
```

Dem Korrektor werden hier (hinter der Überschrift „Teil a)“) zwei Radiobuttons (beschriftet mit „Richtig“ und „Falsch“) angezeigt, der Student sieht je nach vom Korrektor markiertem Radiobutton das Wort „Richtig“ bzw. „Falsch“ (bzw. gar nichts, falls der Korrektor keinen der beiden Radiobuttons markiert hat). Da keine Punkte zugeordnet sind, wird dieser KorrektorCheck nicht zur Berechnung der Gesamtpunktzahl herangezogen.

Die folgenden Beispiele zeigen größtenteils nur noch Varianten des KorrektorCheck-Elements selbst, ohne umgebenden HTML-Text.

```
[KorrektorCheck2 punkte="10" /]
```

Abweichend vom ersten Beispiel ist diesem KorrektorCheck eine Punktzahl zugeordnet. Bei richtiger Antwort wird dem Studenten „Richtig (10 Punkte)“ (statt nur „Richtig“) angezeigt, weiterhin bekommt der Student automatisch 10 Punkte für diese Teilaufgabe vergeben. (Die insgesamt vergebene Punktzahl ist die Summe der Punkte aller als richtig markierten KorrektorChecks.)

```
[KorrektorCheck3 richtig="Ja" falsch="Nein" /]
```

Dieses Beispiel definiert einen KorrektorCheck, dessen Radiobuttons für den Korrektor mit „Ja“ und „Nein“ beschriftet sind, und auch dem Studenten wird „Ja“ bzw. „Nein“ angezeigt. Da keine Punkte zugeordnet sind, wird dieser KorrektorCheck nicht zur Berechnung der Gesamtpunktzahl herangezogen.

```
[KorrektorCheck4 punkte="10" richtig="Ja-" falsch="Nein" /]
```

Hier bekommt der Student bei richtiger Antwort 10 Punkte. Da das richtig-Attribut auf einem ‚-‘ endet, wird ihm bei richtiger Antwort nur „Ja“ (und nicht „Ja (10 Punkte)“) angezeigt.

```
[KorrektorCheck5 punkte="10" richtig="" falsch="" /]
```

In diesem Beispiel sieht der Korrektor zwei Radiobuttons mit den Beschriftungen „Richtig“ und „Falsch“. Der Student dagegen bekommt bei falscher Antwort gar nichts, bei richtiger Antwort die Punktzahl „(10 Punkte)“ angezeigt.

```
[KorrektorCheck6 punkte="10" richtig="-" falsch="" /]
```

Auch in diesem Beispiel sieht der Korrektor zwei Radiobuttons mit den Beschriftungen „Richtig“ und „Falsch“, für den Studenten jedoch bleibt der KorrektorCheck vollständig unsichtbar.

```
<h2>Ihre Einsendung</h2>
$FeldA1
[KorrektorCheck1 punkte="5" block]
```

Dieses Beispiel zeigt eine Einbindung mit `block`-Attribut als eigenen Blockabschnitt, also als eigene Zeile unterhalb der studentischen Einsendung (rechtsbündig).

Beispiele zur Verwendung des `praefix`-Attributs folgen später im Kontext der KorrektorCheckliste.

KorrektorPunkte einbinden

Syntax

Zur Einbindung einer Teilpunkte-Eingabe- oder Auswahlbox in die Korrekturschablone notieren Sie ein Element mit folgendem Aufbau:

- `[KorrektorPunkte`# *Attribute* `/]`

Dabei ist # wieder durch eine Nummer zu ersetzen, welche das KorrektorPunkte-Element eindeutig identifiziert. Der schließende Slash (/) ist beim `KorrektorPunkte`-Element optional (darf also entfallen) – analog zu `KorrektorCheck`, aber anders als bei den `KorrektorKommentar(Inline)`-Elementen, bei denen, falls kein Slash vor der schließenden Klammer des öffnenden Tags steht, ein entsprechendes schließendes Tag am Ende des Vorgabetextes erwartet wird.

Die Attribute beschreiben wir im Folgenden getrennt nach den beiden **Modi**, in denen ein KorrektorPunkte-Element genutzt werden kann:

Eingabefeld-Modus

In diesem Modus wird den Korrektoren ein numerisches Eingabefeld präsentiert, in das sie eine Punktzahl eingeben können. Die Punktzahl darf dabei nicht negativ sein (≥ 0). Optional können Sie dabei per Attribut `max` auch eine Maximalpunktzahl festlegen. Lassen Sie das Attribut weg, so können hier beliebig hohe Punktzahlen eingetragen werden.

Die folgenden optionalen *Attribute* sind in diesem Modus möglich:

Attribut	Bedeutung
`max`	Maximalpunktzahl, die Korrektoren in das Feld eintragen dürfen (s.o.). Beispiel: `max="5"`. Wenn Sie das Attribut nicht angeben, ist eine beliebig große Eingabe ≥ 0 möglich.
`zeigemax`	Flag-Attribut ohne Wertzuweisung. Wenn dieses Attribut *nicht* angegeben wird, sehen die Studierenden in der Korrektur nur, wieviele Punkte ihnen gegeben wurden, z.B. »5 Punkte«. Wenn Sie per `max` eine Maximalpunktzahl festlegen und *zusätzlich* dieses Attribut angeben (z.B. `[KorrektorPunkte1 max="10" zeigemax]`), wird die Anzeige (für Korrektoren und Studierende) um die Anzeige der maximal erreichbaren Punktzahl erweitert, z.B. »5 von 10 Punkten«.
`showmax`	Synonym zu `zeigemax`, beide Schreibweisen sind gleichwertig.
`praefix`	Analog zu KorrektorCheck: Ein Text, der dem Eingabefeld in der Korrektorensicht sowie der Punkteanzeige in der Studentensicht (sofern diese nicht unsichtbar ist) vorangestellt wird. Insbesondere in KorrektorChecklisten sollten Präfixe verwendet werden, um die einzelnen Elemente dieser Liste (z.B. mit verschiedenen Bewertungskriterien) zu beschriften. Für *einzelne* KorrektorPunkte hinter einer studentischen Eingabe wird normalerweise kein Präfix benötigt, da der Bezug i.d.R. klar ist.
`prefix`	Synonym zu `praefix`, beide Schreibweisen sind gleichwertig.
`block`	Analog zu KorrektorCheck: Flag-Attribut ohne Wertzuweisung. Wenn angegeben, wird das KorrektorPunkte-Element als Blockelement (div) gerendert (und per Default-CSS am rechten Seitenrand ausgerichtet), ohne Angabe wird es als Inlineelement (span) eingefügt. Ausnahme: Innerhalb einer KorrektorCheckliste hat dieses Attribut keine Auswirkung.
`value`	Ein Punktewert, der bereits im Feld eingetragen sein soll. Im Normalfall ist von der Angabe dieses Attributs in Korrekturschablonen abzuraten, denn nur, wenn alle KorrektorPunkte zu Beginn unausgefüllt sind (also kein `value`-Attribut besitzen), kann der Korrektor erkennen, welche er schon bearbeitet (ausgefüllt) hat und welche nicht.

Hinweis: Im Input-Modus wird die Maximalpunktzahl (sofern per `max`-Attribut definiert) *in der Korrektorsicht immer* hinter dem Eingabefeld angezeigt. Das Attribut `zeigemax` wirkt sich dann nur auf die Studentensicht aus (vgl. auch Abbildung 3-1, bei der ein Punkteeingabefeld mit `max="5"`, aber *ohne* `zeigemax`-Einstellung eingebunden ist).

Selectbox-Modus

Der Modus zur Einbindung einer Auswahlmöglichkeit (Selectbox) für Korrektoren wird aktiviert, indem Sie zu mindestens einer Punktzahl eine Beschriftung festlegen. In diesem Modus sind alle im vorigen Abschnitt bereits aufgelisteten Attribute weiterhin definiert, und Beschriftungen fügen Sie über Attribute des folgenden Aufbaus hinzu:

Attribut	Bedeutung
`zahl="Text"`	z.B. `15="sehr gut"` legt fest, dass Korrektoren in der Auswahlbox die Option „sehr gut (15 Punkte)“ zur Auswahl bekommen sollen. Falls der Text mit einem Minussymbol – endet, wird die Punktzahl in der Studentensicht nicht mit angezeigt, d.h. bei Auswahl einer Option wie `15="sehr gut-"` sähe der Student dann nur den Text »sehr gut« ohne die nachgestellte Angabe der Punktzahl in Klammern. Falls ein Text mit Minus enden soll (also das Minus angezeigt werden soll und nicht die Punktzahl unterdrücken soll), hängen Sie hinter dem Minus ein Leerzeichen an, z.B. `0="--- "` wird zur Anzeige »--- (0 Punkte)«, während `0="---"` zur Anzeige »--« führen würde.
`max`	Wenn Sie im Selectbox-Modus das Attribut `max` *nicht* angeben, ist automatisch die größte Punktzahl, der Sie eine Beschriftung zuordnen, zu der also ein Attribut der obigen Form angegeben wurde, die Maximalpunktzahl. Korrektoren können dann *nur zwischen den von Ihnen festgelegten Optionen wählen* („diskreter Selectbox-Modus“). Wenn Sie dagegen `max` *zusätzlich* angeben, wird in der Selectbox zu *jeder* Punktzahl von 0 bis zum Maximum eine Auswahloption angeboten, *auch für Punktzahlen ohne Beschriftung* („Range-Modus“). Außerdem wird (Stand Januar 2022) die Optionsliste der Selectbox im Fall der Angabe von `max` immer aufsteigend von 0 bis max sortiert, ohne das Attribut werden Ihre Optionen in genau der Reihenfolge zur Auswahl angeboten, in der Sie die Attribute im Tag notieren.

Punktzahlen mit Nachkommastellen

Seit Herbst 2022 unterstützt das Online-Übungssystem prinzipiell, dass von Korrektoren/-innen vergebene Punktzahlen auch Nachkommastellen haben dürfen, wobei in den Kursparametern einstellbar ist, ob und wie viele (1 oder 2) Nachkommastellen erlaubt sind, siehe: »Punktzahlen mit Nachkommastellen« zu KorrektorChecks.

Wenn Ihr Kurs also ein Punkteformat mit Nachkommastelle(n) verwendet, dann dürfen auch die mit einem KorrektorPunkte-Element zu vergebenden Punkte entsprechend 1 oder 2 Nachkommastellen haben.

Genauer hängt das vom Modus ab:

- Im Eingabefeld-Modus können Korrektoren/-innen Punktzahlen mit Nachkommastellen eingeben, falls sie mehr Nachkommastellen eingeben als in den Kursparametern erlaubt, wird die Eingabe bei Speicherung der Korrektur auf die vorgegebene Nachkommastellenzahl (1 oder 2) gerundet. (Falls in den Kursparametern die Grundeinstellung – nur ganze Punkte / keine Nachkommastellen – aktiv ist, lassen sich nur ganze Zahlen in den Feldern eingeben.)
- Im „diskreten" Selectbox-Modus, in dem nur die von Ihnen vorgegebenen Auswahloptionen (bestimmte Punktzahlen mit Beschriftung) von den Korrektoren/-innen ausgewählt werden können, können Sie hier auch Punktzahlen mit Nachkommastellen vorgeben, z.B. wird mit dem Attribut `2,5="hinreichend"` eine Option mit der Beschriftung „hinreichend" festgelegt, zu der zweieinhalb Punkte vergeben werden. Dabei sind noch folgende Aspekte zu beachten:
  - Die bestmögliche Punktzahl unter ihren Optionen ist auch gleichzeitig die höchste mit dieser KorrekturPunkte-Selectbox zu vergebende Bewertung und kann daher auch Nachkommastellen aufweisen. Wie schon oben im Abschnitt zu Nachkommastellen-Support bei KorrektorChecks ausgeführt, ist dabei aber zu beachten, dass die insgesamt in der Aufgabe erreichbare Punktzahl immer ganzzahlig sein muss.
  - Ebenfalls analog zu KorrektorChecks gilt auch hier, dass, falls Sie Optionen mit mehr Nachkommastellen im HTML vorsehen als laut Kursparametern aktuell erlaubt, diese zum Zeitpunkt der Korrektur auf die maximal erlaubte Nachkommastellenzahl gerundet werden.
- Der „Selectbox-Range-Modus" dagegen, in dem Sie also eine (*ganze*) Maximalpunktzahl angeben und die Korrektoren/-innen jeweils jede Punktzahl von 0 bis zur Maximalpunktzahl auswählen können, wovon Sie ggf. zu ausgewählten dieser Optionen auch einen Beschriftungstext festlegen, funktioniert grundsätzlich nur mit ganzen Zahlen. Die Selectbox enthält eben genau einen Eintrag für jede ganze Zahl von 0 bis zur Maximalzahl, wodurch sich keine Punktzahlen mit Nachkommastellen auswählen lassen.
  - Es ist also insbesondere nicht zulässig, die Angabe einer Maximalpunktzahl (z.B. `max="100"`) mit Teilpunktzahlen mit Nachkommastellen (z.B. `0,5="sehr wenig"`") zu kombinieren. Was genau in solchen Fällen passiert (ob also z.B. eine Range-Auswahl erzeugt wird und die nicht-ganzen Einträge ignoriert werden oder ob das max-Attribut ignoriert wird und eine „diskrete" Auswahl erzeugt wird), ist explizit nicht definiert.

Abschließender Hinweis: Sollen Sie in den Kursparametern die erlaubte Nachkommastellenzahl*nachträglich* reduzieren, nachdem bereits erste Korrekturen von Korrekturkräften erzeugt wurden, so bleiben in diesen Korrekturen die Punktzahlen mit mehr Nachkommastellen unverändert erhalten, die Gesamtpunktzahl wird in den Punkteübersichten oder bei Notenberechung nach Notenschema etc. dann aber später auf die aktuelle Einstellung gerundet.

HTML-Kontext

Hier gilt genau dasselbe wie für `KorrektorCheck`-Elemente, siehe HTML-Kontext KorrektorChecks.

Insbesondere steuern Sie über das `block`-Attribut, ob ein Block- oder Inline-Element erzeugt werden soll, und auch `KorrektorPunkte`-Elemente dürfen Teil einer `KorrektorCheckliste` sein (in welchem Fall das `block`-Attribut ignoriert wird).

Beispiele

```
<p>Ihre Eingabe: $FeldA1 [KorrektorPunkte1]</p>
```

Einfachste Form: Ein Eingabefeld für eine beliebige Punktzahl ≥ 0, ohne vorangestellten Text und als `<span>`-Element (also zur Inline-Anordnung, hier innerhalb eines Absatzes) gerendert.

Dieses Beispiel geht davon aus, dass es sich bei `FeldA1` um ein einfaches Text-Input zur Eingabe z.B. einer Zahl oder eines oder weniger Wörter handelte. Die Eingabe wird als Absatz in die Korrekturseite eingefügt und die Korrekturpunkte im selben Absatz direkt hinter der Eingabe angeordnet.

```
<h2>Ihre Eingabe (HTML-Text)</h2>
$FeldA1
[KorrektorPunkte2 max="15" block]
```

Eingabefeld für eine Punktzahl zwischen 0 und 15 (einschließlich) und als Blockelement (`div`).

Dieses Beispiel geht davon aus, dass `FeldA1` im Aufgabenformular eine Textarea mit WYSIWYG-Editor war, also die Einsendung bereits aus einer Abfolge von HTML-Blockelementen wie Überschriften und Absätzen besteht. Die Punkte werden als eigenes, rechtsbündig ausgerichtetes Blockelement darunter gesetzt.

Die folgenden Beispiele verzichten auf den HTML-Kontext und geben nur noch das Element selbst an.

```
[KorrektorPunkte2 max="15" zeigemax block]
```

Unterscheidet sich vom vorherigen Beispiel dadurch, dass die erreichbare Maximalpunktzahl auch den Studierenden mit angezeigt wird (per » von 15 Punkten«). Für die Korrektoren/-innen ist die Maximalpunktzahl hinter dem Eingabefeld auch im vorherigen Beispiel sichtbar, damit diese besser erkennen können, wieviele Punkte sie maximal vergeben dürfen.

```
[KorrektorPunkte3 max="5" 5="sehr gut-" 4="gut" 1="Ansatz erkennbar" block]
```

Erzeugt für Korrektoren eine Selectbox zur Auswahl von beliebigen Punktzahlen von 0 bis 5, von denen einzelne eine Beschriftung aufweisen, also:

- 0 Punkte
- Ansatz erkennbar (1 Punkt)
- 2 Punkte
- 3 Punkte
- gut (4 Punkte)
- sehr gut

Beachten Sie die Sortierung, dass alle Punkte auswählbar sind, und dass im letzten Fall (5 Punkte) die Anzeige der Punktzahl unterdrückt wurde.

```
[KorrektorPunkte3 max="5" zeigemax 5="sehr gut-" 4="gut" 1="Ansatz erkennbar" block]
```

Variante des obigen Beispiels mit Anzeige der Maximalpunktzahl, also folgenden Optionen:

- 0 von 5 Punkten
- Ansatz erkennbar (1 von 5 Punkten)
- 2 von 5 Punkten
- 3 von 5 Punkten

- gut (4 von 5 Punkten)
- sehr gut

```
[KorrektorPunkte4 praefix="Wertung:" 5="sehr gut" 3="mittel" 0="-" block]
```

Erzeugt in der Korrektorensicht eine Selectbox mit dem vorangestellten Label »Wertung: « und mit genau folgenden Optionen:

- sehr gut (5 Punkte)
- mittel (3 Punkte)
- 0 Punkte

In der Studentensicht wird bei Auswahl einer der beiden ersten Optionen daher entweder »Wertung: sehr gut (5 Punkte)« oder »Wertung: mittel (3 Punkte)« angezeigt. Wählt der Korrektor dagegen »0 Punkte« aus, wird dem/der Studenten/-in überhaupt keine Wertung angezeigt: Die Beschriftung `"-"` legt fest, dass die Punktzahl (0 Punkte) verborgen werden soll und nur der Text vor dem Minus (also hier gar keiner) anzuzeigen ist. Und wenn die Wertung komplett leer ist, wird auch das Präfix nicht vorangestellt. (In der Auswahlbox für den/die Korrektor/-in ist die Auswahloption natürlich nicht leer, sondern dort wird in solchen Fällen die Punktzahl zur Auswahloption angezeigt.)

Prinzipiell können Sie auch *jeder* Punktzahl das Label `"-"` zuordnen. Wenn Sie das sogar für *alle* KorrektorPunkte-Elemente der Aufgabenseite machen (und analog auch für ggf. vorhandene KorrektorChecks jeweils per `richtig="-" falsch=""` die Ausgabe für Studierende unterdrücken), dann können die Korrektoren/-innen in der Korrektur zwar diverse Teilpunktzahlen vergeben, und die Gesamtpunktzahl wird dann automatisch als Summe dieser Teilpunktzahlen berechnet, die Studierenden sehen aber später dann nur die Gesamtwertung. Die Teilwertungen bleiben dann also „intern“, nur für den Korrektor aufgeschlüsselt.

KorrektorCheckliste einbinden

Angenommen, Sie möchten eine ganze Folge von KorrektorPunkte- und/oder KorrektorCheck-Elementen direkt untereinander in die Aufgabenseite einfügen, z.B. als Kriterienkatalog für Teilwertungen zu einer studentischen Einsendung, die wiederum ein längerer Text sein könnte, der in einer WYSIWYG-Textarea eingegeben oder als Datei hochgeladen wurde. Das könnte dann in der Korrekturschablone wie folgt aussehen:

```
<h2>Ihre Einsendung:</h2>
$FeldA1
[KorrektorCheck1 praefix="Maximallänge:" richtig="eingehalten" falsch="überschritten"
punkte="1" block]
[KorrektorPunkte1 praefix="Sprache:" max="5" zeigemax block]
[KorrektorPunkte2 praefix="Stil:" max="1" zeigemax block]
[KorrektorPunkte3 praefix="Rechtschreibung:" max="2" zeigemax block 2="sehr gut"
1="hinreichend"]
… u.s.w.
```

In diesem Fall würden alle diese Elemente als eigene Blockelemente unabhängig voneinander untereinander gesetzt (wie einzelne rechtsbündige Absätze). Wenn aber die Inputs nicht alle gleich breit sind (wie hier, wo Radiobuttons, Selectboxen und Punktzahl-Inputs gemischt wurden), sieht das „unruhig“ und unübersichtlich aus.

Sie können eine Folge von KorrektorChecks und KorrekorPunkten (mit Präfixen) aber einfach mit den Tags `[KorrektorCheckliste]` und `[/KorrektorCheckliste]` (ohne Nummer) umschließen:

```
<h2>Ihre Einsendung:</h2>
$FeldA1
[KorrektorCheckliste]
[KorrektorCheck1 praefix="Maximallänge:" richtig="eingehalten" falsch="überschritten"
punkte="1"]
[KorrektorPunkte1 praefix="Sprache:" max="5" zeigemax]
[KorrektorPunkte2 praefix="Stil:" max="1" zeigemax]
[KorrektorPunkte3 praefix="Rechtschreibung:" max="2" zeigemax 2="sehr gut"
1="hinreichend"]
… u.s.w.
[/KorrektorCheckliste]
```

Die `block`-Attribute dürfen in diesem Fall entfallen, sie werden innerhalb von Checklisten ignoriert.

In dieser Checkliste werden – bei hinreichend breitem Bildschirm – die Elemente tabellarisch ausgerichtet, wobei die Präfixe rechtsbündig, die Wertungen linksbündig stehen. In diesem Beispiel, wo jedes Präfix auf Doppelpunkt endet, stehen also alle Doppelpunkte bündig untereinander (vgl. auch Abbildung 3-1 oben).

Auf schmaleren Bildschirmen (z.B. Smartphones) werden die Präfixe jeweils in einer eigenen Zeile über der Wertung angeordnet und alles linksbündig ausgerichtet.

Auch diese Standarddarstellung der KorrektorChecklisten kann im Zweifel per CSS überstimmt werden, siehe CSS-Anpassung.

Button zum Speichern der Korrektur verändern

Das Online-Übungssystem bindet in der Korrektoransicht einen Button zum Speichern der Korrektur am Ende des HTML-Bodys ein (vgl. z.B. Abbildung 3-1). Dies geschieht vollautomatisch und bedarf keiner Konfiguration von Ihrer Seite.

Falls Sie aber z.B. ein Webdesign für die Korrekturseite verwenden, in dem Sie selbst die Position des Buttons festlegen möchten, den Button vielleicht auch mehrfach an verschiedenen Positionen (z.B. einmal über und einmal unter der Korrektur) einbinden möchten oder falls Sie Formatierungen am Button vornehmen möchten, können Sie optional ein Element `KorrektorSubmit` in die Korrekturschablone aufnehmen.

In der nach dem Speichern einer Korrektur angezeigten Vorschau wird jeweils an Stelle des Speichern-Knopfes ein anderer Button zum Neu-Laden des Korrekturformulars angezeigt. Auch dessen Position wird durch das `KorrektorSubmit`-Element festgelegt.

Syntax

- `[KorrektorSubmit` *Attribute* `/]`

Das Element darf mehrfach im Quelltext vorkommen. Es ist keine identifizierende Nummer (wie bei den anderen Tags) anzugeben. Der abschließende Slash ‚/' ist optional (wie beim KorrektorCheck auch).

Attribute werden 1:1 in HTML-Input-Tag übernommen. Es dürfen alle Attribute angegeben werden, die in HTML-Button-Tags erlaubt sind – mit folgenden Ausnahmen: `type`, `name` und `value`. Insbesondere dürfen Sie auch ein `class`-Attribut hinzufügen. Die Klassennamen, die Sie darin angeben, werden ggf. mit vom Übungssystem intern zugewiesenen Klassen in einem Attribut kombiniert.

HTML-Kontext

Das `KorrektorSubmit`-Element wird direkt durch ein HTML-`button`-Element (inline) ersetzt. Es darf überall im HTML-Code eingebunden werden, wo Formularelemente erlaubt sind. (Sie können voraussetzen, dass der gesamte Inhalt der Korrekturseite ein HTML-Formular ist. Die `form`-Tags werden vom Online-Übungssystem automatisch direkt nach `<body>` bzw. vor `</body>` in die Korrekturseite eingefügt.)

Korrekturansicht für Betreuer

Ein Kursbetreuer kann wie gehabt in der Einsendungs- und Korrekturübersicht die Korrekturen der Studenten einsehen. Er bekommt dabei die Korrektur genauso angezeigt, wie auch der Student sie sehen wird.

Speziell für Betreuer besteht zusätzlich die Möglichkeit, die gespeicherte Korrektur in ihrer *Rohform* einzusehen, also ohne Aufbereitung durch z.B. Ersetzen von Variablen oder optische Aufbereitung der In-Browser-Korrektur-Elemente.

Um diese Rohansicht zu erreichen, kann der Betreuer zunächst eine Korrektur wie gewohnt öffnen und dann in der URL-Zeile seines Browsers hinter dem Servicenamen „KorrekturAccess" den Zusatz „Roh" einfügen. Die resultierende URL sieht dann z.B. folgendermaßen aus:
`https://` *Servername* `/vus/KorrekturAccessRoh/01614/SS11/2/5/7777777/`

CSS-Anpassung

Die Darstellung von In-Browser-Korrektur-Elementen wird durch eine zentrale CSS-Datei bestimmt:
https://online-uebungssystem.fernuni-hagen.de/stylesheets/inBrowserKorrektur.css <https://online-uebungssystem.fernuni-hagen.de/stylesheets/inBrowserKorrektur.css>

Wenn Sie global in Ihrem gesamtem Kurs von diesen Vorgabe-Stilen abweichen möchten, können Sie (seit November 2020) eine eigene Datei gleichen Namens (also `inBrowserKorrektur.css`) in Ihrem Kurs als Kursressource hochladen. Sobald eine Kursressource dieses Namens gefunden wird, wird diese an Stelle der oben genannten zentralen CSS-Datei verwendet.

Prinzipiell haben Sie zumindest in der fortgeschrittenen Aufgabenerstellung natürlich auch noch die Möglichkeit, eigene CSS-Dateien, die unter anderem Namen in den Kursressourcen abgelegt sind, per `<link>`-Tag in Ihre Korrekturseiten einzubinden oder per `<style>`-Element Stile direkt in der Korrekturschablone (HTML-Datei) zu hinterlegen (vgl. das Beispiel im nachfolgenden Unterabschnitt Bearbeitung einer Einsendung im KorrektorKommentar). Beachten Sie dabei aber, dass eine direkte, manuelle Einbindung in der Korrekturschablone (HTML) sich wirklich nur auf die Ansicht der fertigen In-Browser-Korrektur (in der Korrektoren-Sicht also nur auf die Vorschau-Ansicht auswirkt), während die `inBrowserKorrektur.css`-Datei (global oder Ihre Version in den Kursressourcen) zusätzlich auch für die WYSIWYG-Editorboxen für Korrektoren zum Einsatz kommt. Damit kann z.B. eingestellt werden, dass diese Texteditoren schon beim Schreiben dieselbe (per Default rote) Schriftfarbe verwenden, die hinterher auch in der Korrektur-Ansicht verwendet werden wird.
Die Verwendung einer Kursressource `inBrowserKorrektur.css` ist daher ab sofort die empfohlene Variante.

> *Hinweis: Mit dem Update von Ende Januar 2022, das insbesondere KorrektorPunkte und KorrektorChecklisten neu einführte und für KorrektorChecks die Attribute `block` und `praefix` hinzufügte, hat sich auch das Vorgabe-CSS geändert. Falls Sie bereits ein älteres eigenes CSS in den Kursressourcen Ihres Kurses liegen haben, werden sich nicht alle neuen Möglichkeiten wie dokumentiert verhalten. So wird Ihr altes `inBrowserKorrektur.css` z.B. keine Styles für KorrektorChecklisten und KorrektorPunkte enthalten und auch keine Stile für rechtsbündige Ausrichtung einzelner Wertungszeilen (d.h. von KorrektorCheck- und KorrektorPunkte-Elementen mit `block`-Attribut)! Falls Sie die neuen Erweiterungen der In-Browser-Korrektur nutzen möchten und noch ein altes, eigenes CSS installiert haben, sollten Sie sich also die aktuelle Original-CSS herunterladen und Ihre Anpassungen darauf übertragen (und diese neue Fassung dann in den Kursressourcen ablegen)!*

Studentensicht

Die vom Online-Übungssystem zur Darstellung der In-Browser-Korrektur-Elemente in der **Studentensicht** der Korrektur[6] generierten HTML-Elemente können Sie zur Formatierung in CSS wie folgt adressieren:

Korrektor-Element	HTML-Element	CSS-Klasse	ID
`[KorrektorKommentar#]`	`div`	`KorrektorKommentar`	`KorrektorKommentar#`
`[KorrektorKommentarInline#]`	`span`	`KorrektorKommentar`	`KorrektorKommentarInline#`
`[KorrektorCheck#]`	`span`	`KorrektorCheckRichtig` bzw. `KorrektorCheckFalsch`	`KorrektorCheck#`
`[KorrektorPunkte#]`	`span`	`KorrektorPunkte` bei Maximalpunktzahl zusätzlich `richtig`, bei 0 Punkten zusätzlich `falsch`	`KorrektorPunkte#`

Dabei ist wieder jeweils `#` durch eine konkrete Nummer zu ersetzen.

Für `KorrektorCheck`- und `KorrektorPunkte`-Elemente gilt zusätzlich:

- *mit* `block`-Attribut wird zwar die eigentliche Anzeige auch wie oben beschrieben in der Studentensicht als `span` eingefügt, jedoch wird das Ganze zusätzlich eingepackt in ein `div`-Element mit CSS-Klasse `WertungsZeile`.
- Falls ein `praefix`- Attribut angegeben wurde, wird der Präfixtext dem Span vorangestellt.
- Bei Kombination von `praefix`- und `block`-Attribut wird also z.B. für `[KorrektorCheck1 praefix="Präfix:" value="Richtig"]` ein HTML-Element der folgenden Art erzeugt:

```
<div class="WertungsZeile">
 Präfix: <span class="KorrektorCheckRichtig" id="KorrektorCheck1">Richtig</span>
</div>
```

Für `KorrektorCheckliste`-Konstruktionen gilt wiederum Folgendes:

- Für die gesamte Checkliste wird ein `div` der Klasse `KorrektorCheckliste` erzeugt.
- Darin wird für jedes KorrektorCheck- und KorrektorPunkte-Element wiederum ein `div`-Element erzeugt (ohne `class`).
- Innerhalb dieses `div`s wiederum folgen zwei `span`-Elemente, das erste mit `class="prefix"`.
- In das erste der beiden Spans wird der Präfixtext aus dem `prefix`-Attribut eingefügt, bzw. das Span bleibt leer, wenn das prefix-Attribut fehlt.
- Das zweite Span ist dann genau das beschriebene KorrektorCheck- oder KorrektorPunkte-Span-Element mit `class` und `id` wie in obiger Tabelle beschrieben.

Das kann also dann z.B. wie folgt aussehen:

```
<div class="KorrektorCheckliste">
 <div>
  <span class="prefix">Erstes Element in Checkliste:</span>
  <span class="KorrektorPunkte" id="KorrektorPunkte12">1 Punkt</span>
 </div>
 <div>
  <span class="prefix">Zweites Element in Checkliste:</span>
  <span class="KorrektorCheckRichtig" id="KorrektorCheck14">Richtig (1 Punkt)</span>
 </div>
</div>
```

Wenn Sie sich übrigens die beispielhafte KorrektorCheckliste in der Studentensicht aus Abbildung 3-1 ansehen, sehen Sie, dass nach den Default-Styles die Anzeigen für als richtig markierte KorrektorChecks in grün, für als falsch markierte KorrektorChecks in rot und für KorrektorPunkte in blau dargestellt werden. Diese *blaue*

Schriftfarbe für KorrektorPunkte-Wertungen gilt dabei unabhängig von der konkreten Punktzahl, also auch bei 0-Punkte-Wertungen oder Bewertungen mit der maximal erreichbaren Punktzahl.

Es ist aber per CSS z.B. auch möglich, für 0-Punkte-Wertungen eine rote und für Bestwertungen (maximale Punktzahl) eine grüne Schriftfarbe zu wählen, und nur für Punkte dazwischen z.B. beim eingestellten Blau zu bleiben. Dazu dienen die zusätzlichen CSS-Klassen `richtig` bzw. `falsch`, die einem KorrektorPunkte-Span zusätzlich zugeordnet werden, wenn darin eben die Maximalpunktzahl (laut `max`-Attribut) bzw. 0 Punkte eingetragen sind.

Ein solcher Farbcode analog zu den Farben von KorrektorChecks könnte sinnvoll sein, wenn Sie häufig KorrektorChecklisten sowohl aus KorrektorPunkte- als auch KorrektorCheck-Elementen zusammenstellen. Dann wäre alle Maximalwertungen (also korrekte KorrektorChecks und mit vollen Punkten bewerteten KorrektorPunkte) einheitlich grün und alle 0-Punkte-Wertungen einheitlich rot – unabhängig vom Eingabetyp – und nur „Zwischenwerte“ in blau.

Solche CSS-Styles sind in der Vorgabe-CSS-Datei bereits als Kommentar vorbereitet, aber nicht per Default aktiviert.

Korrektorsicht

In der **In-Browser-Korrekturansicht** für Korrektoren werden die Korrektor-Elemente dagegen durch verschiedene Formularelemente ersetzt, die theoretisch ebenfalls per CSS formatiert werden könnten. Eine spezielle CSS-Klasse wird jedoch nur verwendet, falls für `[KorrektorKommentar#]`-Elemente ein WYSIWYG-Editor eingesetzt wird. Dann wird dem `body` des I-Frames des WYSIWYG-Editors die Klasse `ibk` zugewiesen. Somit kann z.B. die Editor-Schriftfarbe des Editors auf dieselbe Schriftfarbe eingestellt werden wie die des `div`-Elements in der späteren Studentensicht, indem der Selektor `body.ibk` verwendet wird (siehe Vorgabe-CSS).

Die oben beschriebenen „Wrapper-Konstrukte“, also einzelne Divs mit `class="Wertungszeile"` oder die Strukturen für KorrektorChecklisten werden in der Korrektorensicht analog verwendet.

Beispiele

Beispiele für CSS-Selektoren:

- `.KorrektorKommentar` selektiert alle Texteingaben eines Korrektors (sowohl in KorrektorKommentar- als auch KorrektorKommentarInline-Elementen).
- `div.KorrektorKommentar` selektiert nur die KorrektorKommentar-Blöcke und
- `span.KorrektorKommentar` selektiert nur die KorrektorKommentarInline-Abschnitte.
- `#KorrektorKommentarInline3` selektiert gezielt das KorrektorKommentarInline-Element mit der Nummer 3.
- `.KorrektorCheckRichtig` selektiert die Textausgabe jedes als richtig markierten KorrektorChecks in der Studentenansicht bzw. die Beschriftung jedes Richtig-Radiobuttons in der Korrektoransicht (z.B. zur Änderung der per Default grünen Schriftfarbe).
- `.KorrektorCheckFalsch` selektiert entsprechend den Text für falsche Antworten.
- `#KorrektorCheck4` selektiert (nur in der Studentenansicht) den Text des KorrektorChecks mit der Nummer 4 (egal ob dieser eine richtige oder falsche Antwort kennzeichnet).

Submit-Button

Falls Sie am „Korrektur speichern“-Knopf CSS-Formatierungen vornehmen möchten, fügen Sie ein `KorrektorSubmit`-Element (siehe oben) ein. In diesem können Sie z.B. per `style`-Attribut Inline-Formatierungen vornehmen oder per `class`-Attribut eine selbst gewählte CSS-Klasse zuordnen und dazu Regeln in die `inBrowserKorrektur.css`-Datei mit aufnehmen.

Der Reload-Button, der anstelle des Speichern-Buttons in der Vorschau nach dem Speichern angezeigt wird, lässt sich hier nicht konfigurieren, aber der hat „von Hause aus“ immer die CSS-Klasse `KorrektorReload`, so dass dazu im Zweifel auch CSS-Regeln definiert werden können.

Dark Mode

Wenn Sie das oben verlinkte Vorgabe-Stylesheet öffnen, werden Sie darin für die verschiedenen Selektoren jeweils noch eine zweite Version für den Dunkelmodus finden. Sofern Sie für Ihre Aufgaben den Dunkelmodus (Dark Mode) nicht explizit unterdrücken (indem Sie entweder ein eigenes Webdesign verwenden, das keinen Dark Mode unterstützt, oder zwar das aktuelle Übungssystem-Webdesign verwenden, in den Aufgaben aber jeweils explizit den Dark-Mode-Support deaktivieren), sollten Sie immer beide Varianten (für Schrift auf weißem Grund im normalen hellen Modus sowie für Schrift auf schwarzem Grund im Dark Mode) bearbeiten.

Die Dark-Mode-Regeln haben derzeit (Stand November 2020) die folgende Form (am Beispiel für eine Regel zu Klasse `KorrektorKommentar`):

```
.KorrektorKommentar {
  color: #8f0000; /* Farbe für Light Mode, also auf hellem Grund */
}
@media (prefers-color-scheme: dark) {
  body.supportsDarkMode:not(.disableDarkMode) .KorrektorKommentar {
    color: #f00; /* Farbe für Dark Mode, also auf dunklem Grund */
  }
}
```

Die äußere Media-Query sorgt dafür, dass die eingefassten Regeln nur angewandt werden, wenn der Benutzer allgemein den Dark Mode in seinem Browser oder Betriebssystem eingestellt hat. Die Eingrenzung auf `body.supportsDarkMode` stellt wiederum sicher, dass diese Regel nur dann zum Einsatz kommt, wenn das HTML-Dokument selbst DarkMode-Kompatibilität deklariert, indem dem Body-Tag die Klasse `supportsDarkMode` zugewiesen wird. Für das aktuelle Übungssystem-Design »Design 2018« ist das der Fall. Nutzen Sie dagegen z.B. noch das ältere, nicht Dark-Mode-kompatible »Design 2010« oder eigene HTML-Seitenvorlagen, die diese Klasse nicht deklarieren, dann sind diese Dark-Mode-Regeln ebenfalls sofort unwirksam und es werden nur die „normalen“ Light-Mode-Styles verwendet. Weiterhin sorgt die Einschränkung `body:not(.disableDarkMode)` bei Verwendung der »Design 2018«-Seitenlayout dafür, dass die Dark-Mode-Regeln ignoriert werden, falls der Benutzer im Seitenfuß die Checkbox »Dunkelmodus unterdrücken« markiert hat.

Mehr zum Thema finden Sie im separaten Handbuch zum Dark Mode <https://online-uebungssystem.fernuni-hagen.de/download/DarkMode/DarkMode.html> .

Bearbeitung einer Einsendung im KorrektorKommentar

Primär ist die In-Browser-Korrektur darauf ausgelegt, dass die Eingaben des Studenten und des Korrektors streng getrennt sind, d.h. der Korrektor sieht die Einsendung des Studenten, kann sie aber nicht modifizieren. Statt dessen kann er an vorgesehenen Stellen in der Korrekturseite seine Kommentare einfügen.

Die herkömmliche Korrektur, bei der ein Korrektor die gesamte Korrekturseite in einem HTML-Editor öffnet, bietet dem Korrektor dagegen mehr Freiheiten, z.B. kann er an beliebigen Stellen in der studentischen Eingabe eigene Anmerkungen einfügen, vorzugsweise in roter Schrift. Er kann auch z.B. Teile der studentischen Einsendung, auf die er Bezug nimmt, hervorheben oder durchstreichen. Beispielsweise bei der Korrektur von Programmieraufgaben ist es praktisch, direkt an den fehlerhaften Stellen im Quellcode des Studenten Korrekturen anbringen zu können und die Fehlerstelle nicht in einem nachfolgenden Kommentar beschreiben zu müssen.

Ein solches Szenario lässt sich auch mit der In-Browser-Korrektur umsetzen. Wie in KorrektorKommentar einbinden beschrieben, kann man für ein KorrektorKommentar-Element einen Inhalt festlegen, und das kann durchaus auch die studentische Einsendung sein.

Das folgende Beispiel stellt eine mögliche Korrekturschablone für eine Programmieraufgabe mit der Möglichkeit zur Korrektur direkt innerhalb des eingesendeten Quelltexts dar:

```
<html>
<head>
 <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">
 <title>Einsendung zu "$Kursname", Aufgabenheft $AufgabenheftNr,
  $Aufgabenname</title>
 <style type="text/css">
  #KorrektorKommentar1 {border:none; color:black}
 </style>
 <!-- WYSIWYG erweitert schriftarten groessenaenderung -->
</head>
<body>
 <h1>Einsendung zu "$Kursname", Aufgabenheft $AufgabenheftNr,
  $Aufgabenname</h1>
 <p>von $Vorname $Nachname</p>
 <p>Korrektor: $Korrektor</p>
 <p>Datum: $KorrekturDatum</p>
 <hr>
 <h2>Aufgabe 8 )</h2>
 <h3>Eingabe in Feld 1:</h3>
 [KorrektorKommentar1 rows="35" cols="100"]<pre><code class="language-
pascal">$FeldA1P</code></pre>[/KorrektorKommentar1]
 <hr>
 <h3>Anmerkungen</h3>
 [KorrektorKommentar2]{zus&auml;tzl. Anmerkungen}[/KorrektorKommentar2]
 <h3>Erreichte Punktzahl: $KorrekturPunkte von $MaximalPunkte</h3>
 <hr>
</body>
</html>
```

Folgende Besonderheiten sind hierbei zu beachten:

1. Unter der Annahme, dass die studentische Einsendung ein Programmquelltext ist, der in einer einfachen Textbox (ohne WYSIWYG-Editor) eingegeben wurde, und dessen Zeilenumbrüche ebenso wie Leerzeichenfolgen (insb. am Zeilenbeginn) unbedingt erhalten bleiben müssen, wird die Variable `$FeldA1P` in ein Element für vorformatierten Text (pre) eingebunden. Sie trägt das Suffix `P`, da die studentische Einsendung selbst keinen HTML-Text, sondern Plaintext enthält und alle darin enthaltenen ‚<'- und ‚>'-Zeichen nicht als HTML-Tagklammern, sondern als Text interpretiert werden sollen.
2. Um den Inhalt des Pre-Elements nicht nur als beliebigen vorformatierten Text auszuzeichnen, sondern als Programm-Quellcode, der vom Übungssystem (in der normalen Anzeige, nicht im WYSIWYG-Editor selbst) automatisch mit Syntax-Hervorhebungen versehen werden soll, wird dieser zusätzlich in ein Code-Element verpackt. Außerdem wird hier die Programmiersprache Pascal explizit per Class-Attribut ausgezeichnet (optional, ohne solche Angabe erfolgt beim Syntax-Highlighting eine automatische Spracherkennung).[7]
3. Das komplette Pre-Element mit der studentischen Einsendung wird als Inhalt in eine große Eingabebox (`KorrektorKommentar1`) eingebunden.
4. Damit der Korrektor nun dort innerhalb des vorformatierten Texts seine eigenen Kommentare einfügen kann, wird für ihn ein HTML-Editor eingebunden (siehe Kommentar im Kopf), und zwar der *erweiterte* Editor, um dem Korrektor zu ermöglichen, seine Bearbeitungen optisch (z.B. durch rote Farbe) von den studentischen Eingaben abzugrenzen.
5. Die Absatzformat-Auswahl des erweiterten Editors wurde absichtlich *nicht* aktiviert. Der gesamte vorformatierte Text, also die gesamte studentische Eingabe, wird vom Editor TinyMCE als zusammenhängender Absatz behandelt, und auch die Enter-Taste fügt innerhalb vorformatierten Textes nur Zeilenumbrüche ein und keine neuen Absätze. Die Absatzformat-Auswahl würde daher stets das Absatzformat für den gesamten Text (und nicht nur für Einfügungen des Korrektors) verstellen – was wiederum zum Verlust der Einrückungen und Zeilenumbrüche der studentischen Eingabe führen würde! Das Absatzformat darf also nicht verstellbar sein. Statt dessen wird dem Korrektor eine Schriftartauswahl

angeboten, so dass er seine Kommentare innerhalb des vorformatierten Textes nicht nur durch rote Schriftfarbe, sondern auch durch Zuweisung einer anderen Schriftart von der studentischen Einsendung abgrenzen kann.

6. Da normalerweise der gesamte KorrektorKommentar in roter Schrift angezeigt wird – hier also auch die studentische Einsendung, was unpassend wäre –, wurde direkt im Kopf der Seite ein CSS-Code zur Überstimmung dieser Regel durch Festlegung schwarzer Schriftfarbe für `KorrektorKommentar1` eingefügt (vgl. Abschnitt CSS-Anpassung). Auch der Rahmen um den leicht schattierten Hintergrund des Korrekturkommentars wird dort abgeschaltet.
   Der Korrektor muss bei dieser Art der Korrektur seine Eingaben wieder manuell hervorheben.

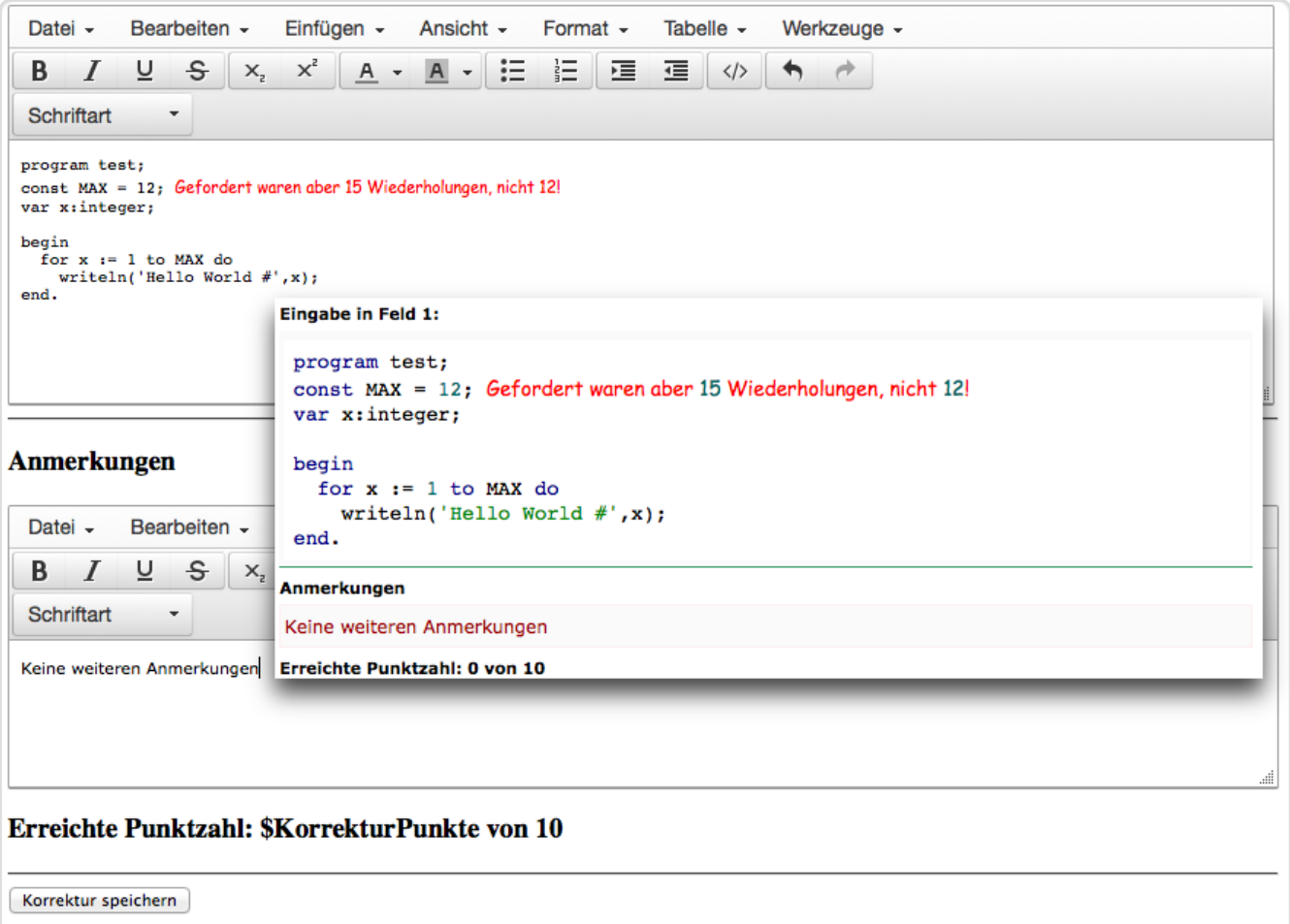


Abb. 3-4: Bearbeitung einer Einsendung im Korrektorkommentar

Obige Abbildung zeigt exemplarisch die In-Browser-Korrektur für obiges Beispiel: Der Korrektor hat einen Kommentar am Ende der zweiten Programmzeile des Studenten angefügt und diese rot eingefärbt sowie die Schriftart „Comic Sans“ gewählt. Ebenfalls in der Abbildung dargestellt ist die Ansicht der Korrektur für den Studenten.

Wichtige Hinweise:

- Bei dieser Konstellation hat der Korrektor natürlich – wie auch bei der klassischen Korrektur im externen HTML-Editor – die Möglichkeit, die studentische Eingabe zu modifizieren und somit auch z.B. versehentlich zu löschen. Die Original-Einsendung des Studenten bleibt aber in der Datenbank des Online-Übungssystems unverändert vorhanden, die Aktionen des Korrektors wirken sich nur auf die Korrektur aus. Der Kursbetreuer kann – genau wie bei herkömmlicher Korrektur auch – in solchen Fällen eine „jungfräuliche" Korrektur wiederherstellen, indem er das Heft des Studenten wieder öffnet und anschließend erneut schließt. Dabei werden jedoch alle Korrekturen für diesen Studenten des gesamten Aufgabenhefts (nicht nur gezielt einer Aufgabe) neu erstellt, entsprechende Korrekturen für diesen Studenten in diesem Aufgabenheft werden also gelöscht. Falls neben der „beschädigten" Korrektur bereits andere Korrekturen für denselben Studenten im selben Aufgabenheft fertiggestellt waren, sollte sich der Korrektor letztere zunächst lokal als HTML-Dateien sichern, bevor der Kursbetreuer das Heft wieder öffnet.
- Der Korrektor kann bei Fehlbedienungen im TinyMCE-Editor in der Regel die Undo-Funktion des Editors benutzen, um die Fehlbedienung rückgängig zu machen.

In-Browser-Korrektur und Offline-Korrektur

Die In-Browser-Korrektur ist ausschließlich bei Online-Korrektur verfügbar und nicht mit dem Offline-Korrekturassistenten kompatibel!

Sollte ein Korrektor sich die ihm zugeteilten Korrekturen mit dem Korrekturassistenten herunterladen, kann er sie dennoch korrigieren, jedoch nicht im Browser, sondern auf die herkömmliche Art mit externem HTML-Editor[8]. Er wird dann in der Korrektur den in der Aufgabenschablone zur Definition der In-Browser-Korrektur-Elemente verwendeten Markup-Code sehen, kann dort jedoch prinzipiell auch seine Kommentare hineinschreiben. Nachfolgende Abbildung zeigt beispielhaft die herkömmliche Offline-Korrektur für das Fallbeispiel aus Abschnitt „Bearbeitung einer Einsendung im KorrektorKommentar".

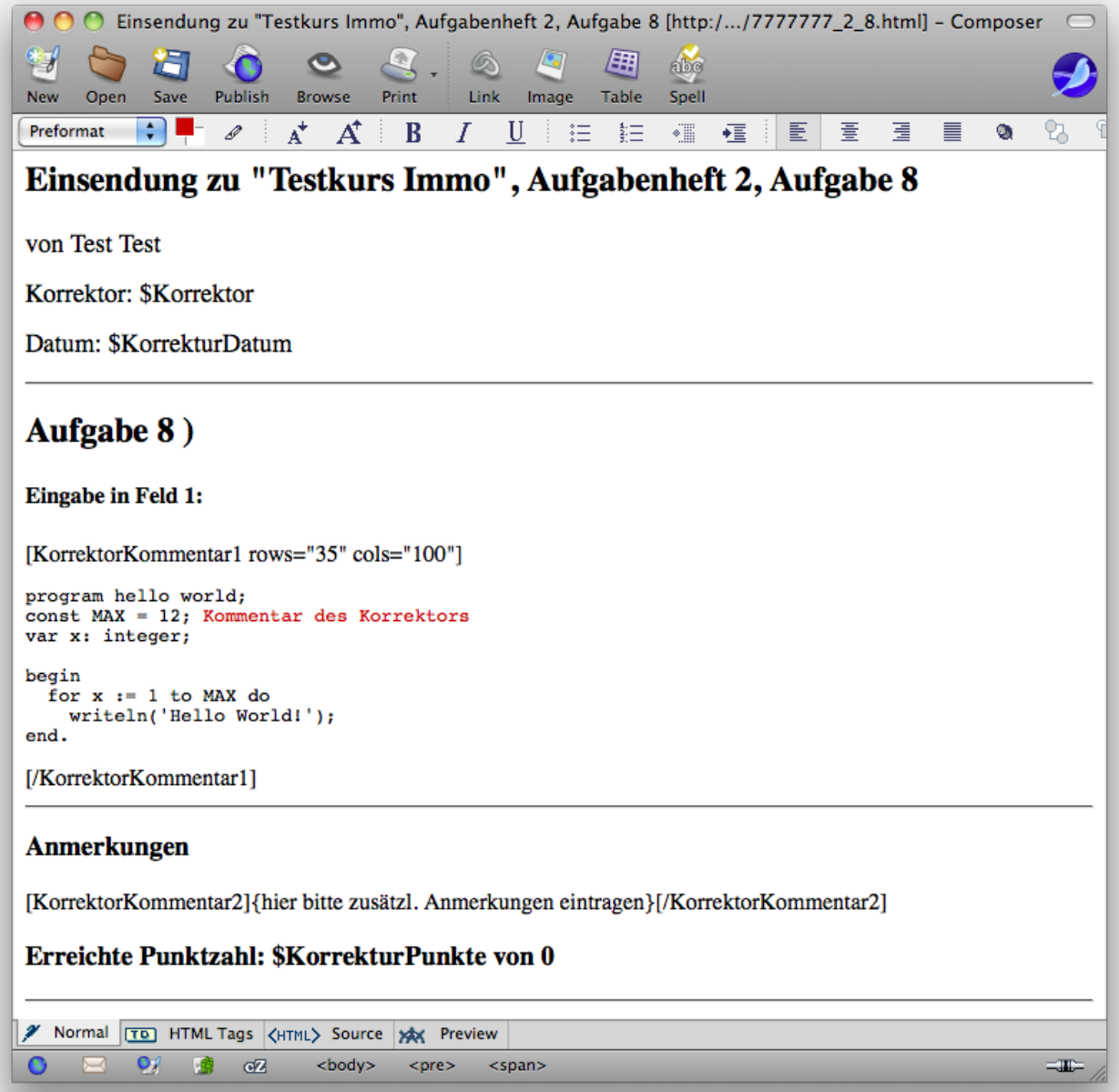


Abb. 3-6: KorrektorKommentare bei Offline-Korrektur

So lange alle KorrektorKommentar-Elemente aus einem öffnenden und einem schließenden Tag bestehen (wie in obiger Abbildung), kann der Korrektor bei der Offline-Korrektur seine Anmerkungen ganz einfach direkt zwischen diese für ihn sichtbaren Tags schreiben, und die Korrektur wird später für den Studenten genauso aussehen, als sei sie mit In-Browser-Korrektur erstellt worden. Problematischer ist es, falls atomare Tags wie `[KorrektorKommentar5 /]` vorkommen. Hier müsste der Korrektor den Querstich aus dem Tag löschen, seinen Kommentar dahinter schreiben und selbst das schließende Tag `[/KorrektorKommentar5]` hinter seinen Kommentar setzen. Falls also die Offline-Korrektur unterstützt werden soll, ist es besser, wenn der Aufgabenautor statt atomarer Elemente stets ein öffnendes und ein schließendes Tag (mit leerem Inhalt dazwischen) in die Korrekturschablone einfügt, z.B.: `[KorrektorKommentar5][/KorrektorKommentar5]`.

Alternativ kann der Offline-Korrektor natürlich auch die KorrektorKommentar-Elemente ganz löschen und seine Anmerkungen in herkömmlicher Weise einfügen und selbst rot einfärben.

Sollten in der Korrekturschablone nicht nur KorrektorKommentare, sondern auch KorrektorChecks vorkommen, so kann ein Offline-Korrektor diese ggf. ausfüllen, indem er im KorrektorCheck-Tag das Attribut `value` einfügt und ihm den Wert `"Richtig"` bzw. `"Falsch"` zuweist. Analog könnte er zu KorrektorPunkte-Elementen

ein Attribut `value` einfügen und darin die von ihm vergebene Punktzahl eintragen (z.B. `value="7"`). Das ist aber unkomfortabel und fehleranfällig, außerdem wird auch bei dieser Art der Korrektur keine automatische Berechnung der erreichten Gesamtpunktzahl (als Summe der Punkte aller richtigen KorrektorChecks) durchgeführt. Es empfiehlt sich, Korrekturen von KorrektorChecks oder KorrektorPunkten ausschließlich in der Online-Korrektur durchzuführen.

Syntax-Highlighting und Quelltext-Einsendungen

Automatisches Syntax-Highlighting einbinden

Das Online-Übungssystem bindet ggf. automatisch einen JavaScript-basierten Syntax-Highlighter („highlight.js“) ein, der Schlüsselwörter, Kommentare, Literale etc. in Sourcecodes automatisch einfärbt, vgl. z.B. Abbildung 3-4.

Assistentengestützte Einbindung

Wenn Sie Ihre Aufgabenseite mit einem Aufgabenerstellungs-Assistenten des Online-Übungssystems erstellen, können Sie seit Mai 2014 die „Code einfügen/bearbeiten“-Funktion des WYSIWYG-Editors verwenden, um z.B. Quelltexte in Ihren Aufgabentext einzufügen. Details siehe im folgenden Abschnitt Studentische Einsendungen in WYSIWYG-HTML-Editoren.

Die Einbindung von Syntax-Highlighting für studentische Eingaben behandelt ein späterer Abschnitt.

Manuelle Einbindung in HTML-Quellcode

Die Einbindung des Syntax Highlighters erfolgt automatisch dann, wenn in einer Aufgaben- oder Musterlösungsseite (auch in einer Quittung oder Korrektur, siehe folgenden Abschnitt) ein Code-Block innerhalb eines Pre-Blocks gefunden wird, d.h. die einfachste Syntax zur Einbindung sieht wie folgt aus:

`<pre><code>`*Der Quellcode*`</code></pre>`

Bei dieser Schreibweise versucht der Syntax Highlighter heuristisch, die Sprache zu erraten und ein passendes Farbschema dazu auszuwählen. Alternativ kann auch eine Sprache explizit deklariert werden, z.B.:

`<pre><code class="language-java">`*Java-Quellcode*`</code></pre>`

Diese Darstellung ist konform zu den W3C-Empfehlungen für HTML 5. Pre-Blöcke ohne Code-Block werden ebenso wenig formatiert wie Code-Blöcke, die nicht innerhalb von Pre-Blöcken liegen.

Die folgende Tabelle gibt den derzeitigen Stand (subject to change) der unterstützten Sprachen und der dazu im `class`-Attribut zu notierenden Klasse wieder:

Sprache	CSS-Klasse
AppleScript	`language-applescript`
Backus-Naur-Form (BNF)	`language-bnf`
Bash/shell	`language-bash`
Brainfuck	`language-brainfuck`
C++	`language-cpp`
C#	`language-cs`
CoffeeScript	`language-coffeescript`
CSS	`language-css`
D	`language-d`
Delphi / Pascal	`language-delphi`
Erlang	`language-erlang`
F#	`language-fsharp`
Go	`language-go`
Haml	`language-hall`
Haskell	`language-haskell`
HTML	`language-html`
Java	`language-java`
JavaScript	`language-js`
JSON	`language-json`
Lasso	`language-lasso`
Less CSS	`language-less`
Lisp	`language-lisp`
LiveScript	`language-livescript`
Lua	`language-lua`
Markdown	`language-markdown`
Mathematica	`language-mathematica`
Matlab	`language-matlab`
Objective-C	`language-objectivec`
Perl	`language-perl`
PHP	`language-php`
Prolog	`language-prolog`
Python	`language-python`
R	`language-r`
Ruby	`language-ruby`
Scala	`language-scala`
Scheme	`language-scheme`
SCSS	`language-scss`
Smalltalk	`language-smalltalk`
SQL	`language-sql`
TeX	`language-tex`
VB.Net	`language-vbnet`
VBScript	`language-vbscript`
XML	`language-xml`

Um für einen bestimmten Code-Block das automatische Syntax-Highlighting zu verhindern, verwenden Sie die Klasse `no-highlight`:

```
<pre><code class="no-highlight">Hier erfolgt kein Syntax Highlighting</code></pre>
```

Highlight-Stylesheet

Der Syntax Highlighter bietet eine ganze Reihe von Stylesheets (i.W. Farbschemata) zur Quelltextauszeichnung an. Das Übungssystem verwendet ein bestimmtes Schema als Default (derzeit das an Google Code angelehnte Schema, aber der Default könnte sich in Zukunft auch ändern).

Falls Sie selbst für Ihren Kurs ein ganz bestimmtes Schema einstellen möchten, finden Sie in den Kursparametern eine entsprechende Auswahlmöglichkeit. Diese wählt das Stylesheet global für alle Seiten Ihres Kurses.

Quelltexte in studentischen Einsendungen

Während Sie in Aufgabentexten oder Musterlösungen eigene Code-Fragmente problemlos wie oben beschrieben auszeichnen können, gibt es bei studentischen Code-Einsendungen verschiedene Dinge zu beachten. Im Wesentlichen haben Sie hier zwei Alternativen:

Studentische Einsendungen in Plaintext-Textareas

Dies ist normalerweise die empfohlene Art, Aufgabenformulare zur Codeeinsendung zu konfigurieren: Studenten bekommen eine Textarea für Plain-Text, d.h. *nicht* mit HTML-Editor ausgestattet, angezeigt und die gesamte Eingabe dieser Textarea soll als Quelltext interpretiert werden. (Dies ist insbesondere auch dann sinnvollste Eingabeart, wenn Sie ein eigenes Vorkorrekturmodul zur automatische Auswertung des eingegebenen Codes einsetzen möchten.)

Assistentengestützte Einbindung

Erstellen Sie Ihre Aufgaben mit dem Erstellungsassistenten für handbewertete Aufgaben, so fügen Sie Ihrer Aufgabe ein Element des Typs »Quelltext-Eingabebox« hinzu. Das erzeugt eine Textarea für Quelltext und konfiguriert die Darstellung in Quittungs- und Korrekturseite zur Verwendung von Syntax-Highlighting. Die Sprache können Sie direkt im Assistenten auswählen.

Fortgeschrittene Aufgabenerstellung

In diesem Fall ist die studentische Einsendung in Quittungs- und Korrekturseite typischerweise wie folgt einzubinden:

```
<pre><code class="language-java">$FeldA1P</code></pre>
```

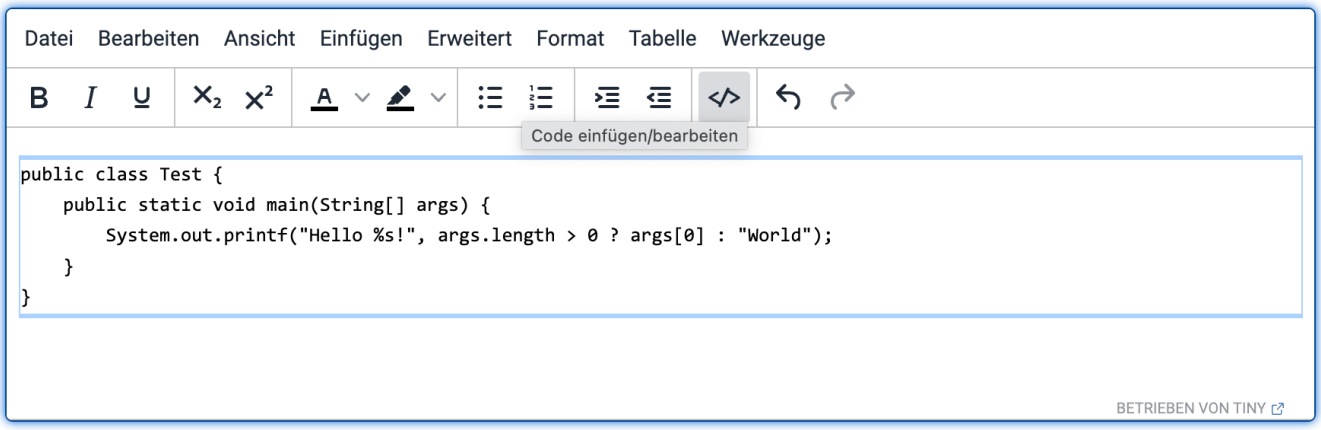
(In diesem Beispiel ist natürlich das Class-Attribut auf die in der Aufgabe erwartete Sprache einzustellen – oder ganz wegzulassen – und die FeldNummer (hier `A1`) ggf. anzupassen, falls die einzufügende Eingabe nicht dem ersten Feld des Aufgabenformulars entnommen werden soll.)

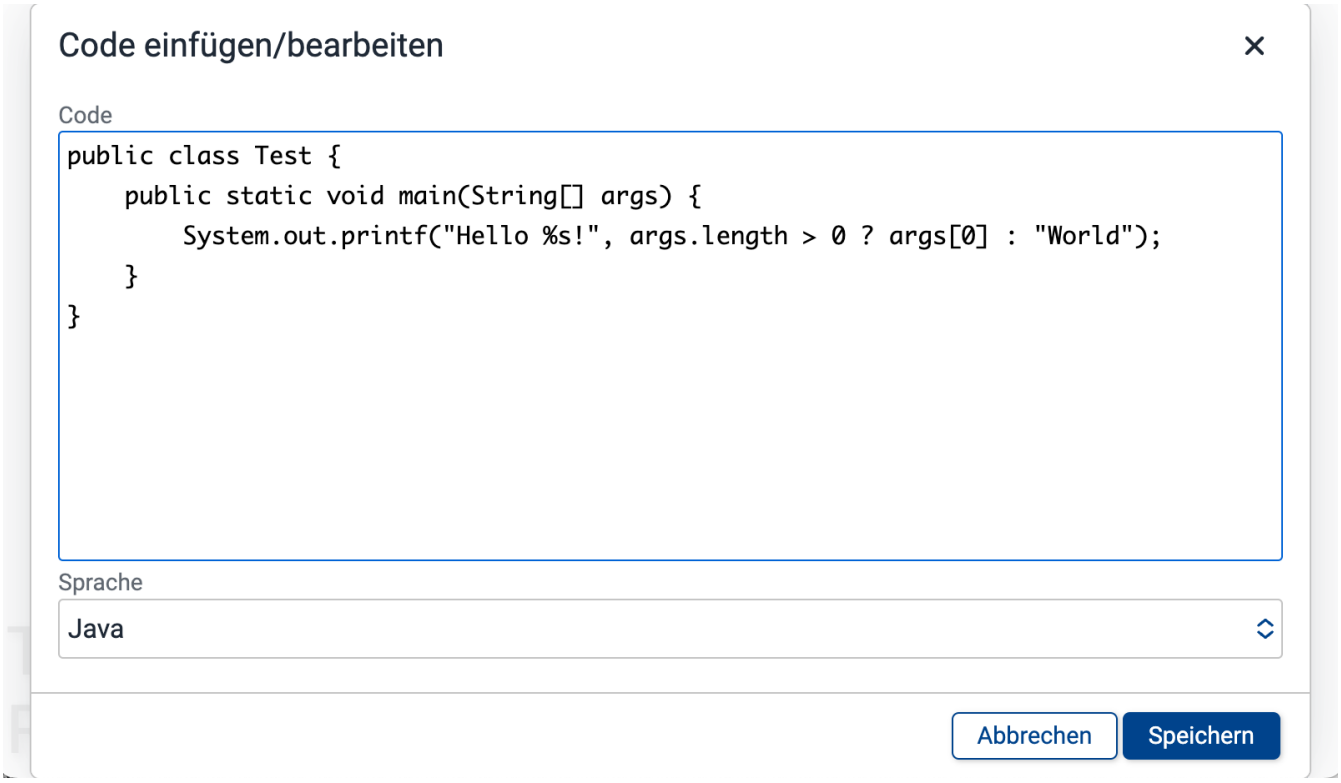
Das `P`-Suffix am Feldnamen ist wichtig, damit der eingesendete Text nicht als HTML-, sondern als Plaintext interpretiert wird. Siehe auch das Beispiel und die Erläuterungen aus Abschnitt Bearbeitung einer Einsendung im KorrektorKommentar.

Studentische Einsendungen in WYSIWYG-HTML-Editoren

Falls Sie nicht eine einzelne Eingabebox nur für Code (und ggf. zusätzliche für Text/Erläuterungen) ins Aufgabenformular einbinden möchten, sondern den Studenten eine Eingabemöglichkeit bieten wollen, in der jeder Student eine beliebige Abfolge von Klartext-Absätzen und Quellcode, ggf. noch strukturiert durch Überschriften – sprich: ein Dokument mit ggf. Quelltext-Anteilen – eingeben können soll, dann verwenden Sie eine Textarea und kombinieren Sie diese mit einem WYSIWYG-Editor, wie im zweiten Kapitel genauer beschrieben.

Im neuen TinyMCE-4-Editor steht Studenten zu diesem Zweck eine spezielle Funktion „Code einfügen/bearbeiten“ zur Auswahl. Damit können sie Quellcode z.B. aus einer IDE direkt einfügen und optional auch die jeweilige Programmiersprache auswählen – voreingestellt ist „Sprache automatisch erkennen“:





Die so eingefügten Code-Teile werden automatisch so ausgezeichnet, wie weiter oben in diesem Kapitel beschrieben, erhalten also auch automatisch das korrekte Syntax Highlighting. Wichtig ist lediglich, dass die `$Feld`-Variable – wie schon oben im Kapitel zur WYSIWYG-Editor-Einbindung beschrieben – ohne P-Suffix und in einem wie oben beschriebenen HTML-Kontext in die Quittungs- und Korrekturseite eingebunden wird (bei assistentengestützter Aufgabenerstellung erfolgt dies automatisch passend).

Fußnoten

1. Die Eingabe von Formeln ist prinzipiell in allen WYSIWYG-Editoren, auch dem kompakten, ohne Plugin direkt im Fließtext möglich, indem dort LaTeX-Code eingegeben wird, umschlossen wahlweise von `$[` und `$]` (oder alternativ links und rechts jeweils `$$`) für abgesetzte Formeln oder von `$(` und `$)` für Formeln innerhalb von Textabsätzen. Das Editorplugin (Wurzelsymbol in der Symbolleiste) lädt aber optional einen Formeleditor zur interaktiven Eingabe oder Nachbearbeitung einer solchen Formel. »WYSIWYG« (also »What you see is what you get«) gilt hier übrigens nur innerhalb des Formeleditorfensters. In der Texteditor-Box wird nur der LaTeX-Code dargestellt (und ist dort auch noch editierbar). Erst nach Einsendung bzw. Speicherung der Eingaben erfolgt dann in der Anzeige des erzeugten Textes innerhalb des Online-Übungssystems wieder auch ein Formel-Rendering.
2. Streng genommen bietet zumindest der aktuelle Editor (TinyMCE seit Version 4) in seiner erweiterten Form *immer* die Möglichkeiten, Absatz- und Schriftformatierungen anzuwenden, und zwar über die Menüleiste. Die optionalen Einstellungen bewirken nur, dass eine weitere Symbolleiste die Absatzformat-, Schriftart- oder Schriftgrößen-Auswahl prominenter und schneller zugänglich anbietet, vgl. Abb. 2-3.
3. Bild-Einbettung in HTML-Code erfolgt in Base64 <http://de.wikipedia.org/wiki/Base64> -Kodierung mit entsprechend größerem Speicherbedarf.
4. Der Wort- / Zeichenzähler steht im Übungssystem erst seit der Umstellung auf TinyMCE 5 im Januar 2023 zur Verfügung. Falls Sie das Einbinden eines älteren Editors erzwingen, fehlt dieser Zähler in der Statusleiste.
5. Die Option `keinebilder` funktioniert *nicht* im alten Editor TinyMCE 3, d.h. falls Sie die Nutzung von TinyMCE 3 erzwingen, bleibt die Option wirkungslos.
6. Mit Studentensicht der Korrektur ist die fertige Darstellung gemeint, wie sie insbesondere dem Studenten nach Korrekturfreigabe angezeigt wird. Korrektoren können diese Ansicht ebenfalls als „Vorschau“ abrufen, und Betreuer sehen beim Ansehen einer Korrektur auch dieselbe Ansicht.
7. Hinweis: Diese Verschachtelung eines Code- innerhalb eines Pre-Elements darf nicht verwendet werden, wenn Sie die Verwendung des alten Editors TinyMCE 3 erzwingen, denn der kann dieses Konstrukt nicht verlustfrei bearbeiten!
8. Bei Online-Korrektur ist das im Übrigen nicht möglich: Ein Upload mit dem Composer wird dort fehlschlagen, wenn die Aufgabe KorrektorKommentar- oder KorrektorCheck-Elemente enthält, also klar auf In-Browser-Korrektur ausgelegt ist.