# Hausaufgabe 3: CSS-Positionierung

Laden Sie ihre Lösung bis zum 19. Dezember um 23:59 Uhr als ZIP-Archiv bei ISIS hoch. Die Ordnerstruktur des ZIP-Archivs soll dabei genau der Struktur der Vorgabe entsprechen. Es dürfen keine Dateien umbenannt, hinzugefügt oder entfernt werden.

## 3.1 JavaScript (8 Punkte)

In dieser Aufgabe beschäftigen wir uns mit Funktionen höherer Ordnung und Lambda-Ausdrücken. Hierzu wollen wir zwei Datensätze verarbeiten, die eine Studierenden- und Prüfungsverwaltung nachbilden. Sie finden diese Datensätze in der Vorlage unter `ha03/js/db/`.

Leider sind diese Datensätze nicht frei von Fehlern und Unschlüssigkeiten. So gibt es zum einen mehrere Studierende mit identischen Matrikelnummern, zum anderen gibt es Studierende, die mehrere Noten in ein und dem selben Kurs haben. Es gibt aber auch ganz banale Hürden zu überwinden, wie die Verwendung von Kommas als Dezimaltrennzeichen. Ihre Aufgabe ist es im Folgenden, einen Überblick über die Daten zu schaffen und die zuvor genannten fehlerhaften Einträge zu identifizieren.

In dieser Aufgabe sollen Sie über Arrays iterieren, indem Sie `map()`, `filter()` und `reduce()` verwenden. Implementieren Sie **keine for-Schleifen**. Bearbeiten Sie für diese Aufgabe ausschließlich die Datei `index.js`!

Ihnen sind zusätzlich in der Vorgabe für jede Teilaufgabe die vollständigen resultierenden Arrays in der geforderten Sortierung als JSON gegeben, damit Sie Ihre Lösung abgleichen können. Sie finden diese Arrays jeweils in den Dateien `js-a0x-result.json`.

### 3.1.1 notGoodGrades (1 Punkt)

Betrachten Sie die Datei `index.js`.

Implementieren Sie die Funktion `notGoodGrades()`, welche die Variable `grades` als Array (siehe `db/grades.js`) übergeben bekommt und nach allen Noten filtert, die eine 3,0 oder schlechter sind.

**Hinweis:** Beachten Sie, dass im Datensatz die Noten als String repräsentiert sind. Außerdem wurden Kommas als Dezimaltrennzeichen verwendet.

### 3.1.2 gradeOverview (1 Punkt)

Betrachten Sie die Datei `index.js`.

Implementieren Sie die Funktion `gradeOverview()`, welche die Variablen `students` und `grades` übergeben bekommt und für jede*n Studierende*n eine Notenübersicht erstellt. Dabei soll jedes Element im `students`-Array auf ein Objekt im folgenden Format abgebildet werden: `student: (students[i]), grades: [(grades[j], grades[j+k], ...)]` .

### 3.1.3 duplicateStudents (2 Punkte)

Betrachten Sie die Datei `index.js`.

Implementieren Sie die Funktion `duplicateStudents()`, welche die Variable `students` übergeben bekommt und nach Studierenden mit der selben Matrikelnummer filtert.

**Tipp:** Bilden Sie zuerst alle Elemente in `students` auf ihre `matrikelnummer` ab. Danach können Sie relativ einfach nach Duplikaten filtern. Bilden Sie das Resultat abschließend auf folgendes Format ab: `matrikelnummer: (matrikelnummer), students: [(students[i], students[j], ...)]` .

**Hinweis:** Sie sollen nach Duplikaten im Array `students` filtern. Das schließt nicht aus, dass es Duplikate im resultierenden Array gibt. Duplikate im resultierenden Array sollen nicht betrachtet werden, d.h. sie werden ebenfalls in das Zielformat abgebildet.

### 3.1.4 invalidGrades (4 Punkte)

Betrachten Sie die Datei `index.js`.

Implementieren Sie die Funktion `invalidGrades()`, welche die Variable `grades` übergeben bekommt und nach *potentiell* fehlerhaften Noten filtert, indem sie ermittelt, für welche Matrikelnummern mehrere Noten für den selben Kurs übermittelt wurden.

**Beispiel:** Für Matrikelnummer X wurde für Kurs Y sowohl eine 2,7 als auch eine 2,3 übermittelt, was ungültig ist. Ihre Lösung soll jedoch auch ”valide“ Duplikate ausgeben, z.B. wenn ein Teilnehmer mit einer 5,0 durchgefallen ist und dann im zweiten Versuch eine 2,3 erhalten hat.

**Tipp:** Beachten Sie ganz genau die Vorgabe hierzu.

## 3.2 CSS (12 Punkte)

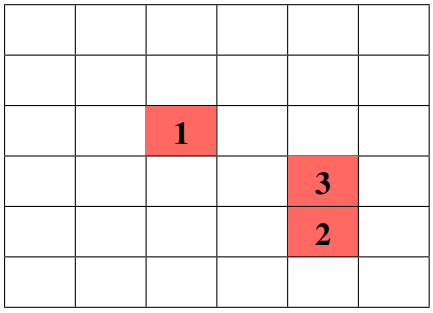
Wir befassen uns nun mit der Positionierung von Elementen mittels CSS. Ihnen ist hierzu im Ordner `ha03/css/` der Vorgabe die Datei `positioning.css` gegeben. Wenn Sie diese in Ihrem Browser öffnen, sehen Sie für die Unteraufgaben 3.2.1, 3.2.2 und 3.2.3 jeweils ein Raster, in dem drei Kästchen rot hervorgehoben werden. In den Rastern hat jedes Kästchen eine Größe von genau 50 x 50 Pixeln.

Ordnen Sie die rot hervorgehobenen Elemente entsprechend der Anweisungen in der jeweiligen Unteraufgabe an.

**Hinweis:** Da die Raster lediglich Hintergrundbilder sind, kann es passieren, dass sie mit leichten Ungenauigkeiten gerendert werden. Sie dienen daher nur der Orientierung. Unabhängig hiervon sollen die Positionen der roten Elemente strikt im 50-Pixel Raster angeordnet werden (die x- und y-Koordinaten sollen also ein Vielfaches von 50 sein).

### 3.2.1 Positionierung absolut (3 Punkte)

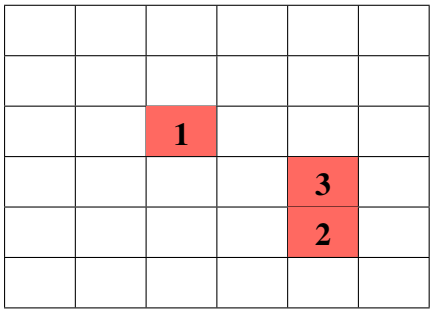
Ergänzen Sie für Ihre Lösung ausschließlich die Regeln `a11`, `a12` und `a13` der CSS-Datei `ha03/css/style.css`.




Verwenden Sie **absolute Positionierung**, um die rot hervorgehobenen Kästchen 1, 2 und 3 genau wie in der Abbildung dargestellt zu platzieren.

### 3.2.2 Positionierung relative (3 Punkte)

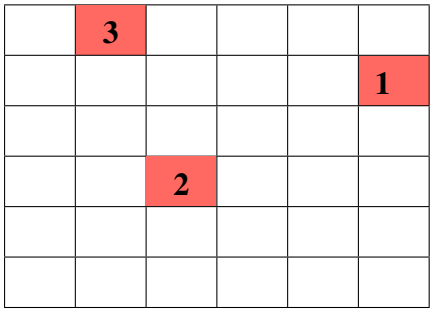
Ergänzen Sie für Ihre Lösung ausschließlich die Regeln `a21`, `a22` und `a23` der CSS-Datei `ha03/css/style.css`.




Verwenden Sie **relative Positionierung**, um die rot hervorgehobenen Kästchen 1, 2 und 3 genau wie in der Abbildung dargestellt zu platzieren.

### 3.2.3 CSS-Grid-Layout (3 Punkte)

Ergänzen Sie für Ihre Lösung ausschließlich die Regeln `wrapper`, `a31`, `a32` und `a33` der CSS-Datei `ha03/css/style.css`.




Verwenden Sie das **CSS-Grid-Layout**, um die rot hervorgehobenen Kästchen 1, 2 und 3 genau wie in der Abbildung dargestellt zu platzieren. Definieren Sie dabei Ihr Grid so, dass es exakt dem oben beschriebenen 50-Pixel Raster entspricht.

### 3.2.4 Responsive Design mit Bootstrap (3 Punkte)

Ergänzen Sie für Ihre Lösung ausschließlich die Elemente mit den IDs `a324`, `a3241`, `a3242` und `a3243` in der HTML-Datei `ha03/css/positioning.html`. Fügen Sie diesen Elementen weitere **Bootstrap-Klassen** hinzu, um ein **responsive Design** zu realisieren. Sie dürfen

auf keinen Fall die vorgegebenen IDs und Klassen der Elemente verändern. Die Elemente dürfen mehrere CSS-Klassen haben.

Aktuell werden die Karten der drei vorhergegangenen Unteraufgaben immer untereinander angeordnet. Für das responsive Design ihrer Lösung soll jedoch folgendes gelten:

- Ist der Bildschirm mindestens 992 Pixel breit, sollen all drei Karten nebeneinander (also in drei Spalten) angeordnet werden.
- Bei einer Bildschirmbreite zwischen 991 Pixeln und 768 Pixeln sollen die Karten auf zwei Spalten aufgeteilt werden.
- Bei Bildschirmbreiten kleiner als 768 Pixeln sollen die Karten (wie bisher) in nur einer Spalte untereinander angeordnet werden.

**Hinweis:** Die Werte der o.g. Anforderungen entsprechen den von Bootstrap definierten Break-Points.