

<https://github.com/openHPI/codeocean/tree/master/app/assets/stylesheets>

Kannst du mir sagen, wozu jedes Stylesheet dient?

<https://github.com/openHPI/codeocean/tree/master/app/assets/stylesheets>

Hier ist eine kurze Zusammenfassung dessen, wofür jedes Stylesheet in diesem Verzeichnis gedacht ist:

- `application.scss`: Dies ist das Haupt-Stylesheet für die Anwendung und enthält alle allgemeinen Stile für die gesamte Website.
- `auth.scss`: Dieses Stylesheet enthält spezifische Stile für die Authentifizierungsseiten der Anwendung.
- `courses.scss`: Dieses Stylesheet enthält spezifische Stile für die Seiten, die mit Kursen und Lektionen in der Anwendung zu tun haben.
- `dashboard.scss`: Dieses Stylesheet enthält spezifische Stile für die Dashboard-Seite der Anwendung.
- `editor.scss`: Dieses Stylesheet enthält spezifische Stile für den Code-Editor der Anwendung.
- `errors.scss`: Dieses Stylesheet enthält spezifische Stile für die Fehlerseiten der Anwendung.
- `landing.scss`: Dieses Stylesheet enthält spezifische Stile für die Startseite der Anwendung.
- `pages.scss`: Dieses Stylesheet enthält spezifische Stile für verschiedene Seiten der Anwendung, die nicht von den anderen Stylesheets abgedeckt werden.
- `profile.scss`: Dieses Stylesheet enthält spezifische Stile für die Profilseite der Anwendung.
- `search.scss`: Dieses Stylesheet enthält spezifische Stile für die Suchseite der Anwendung.
- `sessions.scss`: Dieses Stylesheet enthält spezifische Stile für die Sitzungsseiten der Anwendung.
- `shared.scss`: Dieses Stylesheet enthält gemeinsame Stile, die auf mehreren Seiten der Anwendung verwendet werden.
- `sidebar.scss`: Dieses Stylesheet enthält spezifische Stile für die Seitenleiste der Anwendung.
- `teams.scss`: Dieses Stylesheet enthält spezifische Stile für die Seiten, die mit Teams in der Anwendung zu tun haben.
- `users.scss`: Dieses Stylesheet enthält spezifische Stile für die Seiten, die mit Benutzerkonten in der Anwendung zu tun haben.

application.css:

```
arduino
/*
 * This is a manifest file that'll be compiled into application.css, which
will include all the files
 * listed below.
 */
```

Dieser Kommentar erklärt, dass dieser Dateiinhalt in die `application.css`-Datei kompiliert wird und alle unten aufgeführten Dateien enthält.

```
css
/*
 * Any CSS and SCSS file within this directory, lib/assets/stylesheets,
vendor/assets/stylesheets,
 * or vendor/assets/stylesheets of plugins, if any, can be referenced here
using a relative path.
 */
```

Dieser Kommentar beschreibt, dass jede CSS- oder SCSS-Datei in diesem Verzeichnis oder einem der angegebenen Pfade (`lib/assets/stylesheets`, `vendor/assets/stylesheets` oder `vendor/assets/stylesheets` von Plugins) mit einem relativen Pfadreferenz hier referenziert werden kann.

```
css
/*
 * You're free to add application-wide styles to this file and they'll
appear at the bottom of the
 * compiled file so the styles you add here take precedence over styles
defined in any styles
 * defined in the other CSS/SCSS files in this directory. It is generally
better to create a new
 * file per style scope.
 */
```

Dieser Kommentar beschreibt, dass die Datei für anwendungsbreite Stile genutzt werden kann, die am Ende der kompilierten Datei erscheinen und dadurch Vorrang gegenüber anderen Stilen haben, die in anderen Dateien in diesem Verzeichnis definiert sind. Es ist jedoch besser, für jeden Stilbereich eine neue Datei zu erstellen.

```
javascript
*= require pagedown_bootstrap
```

Diese Anweisung lädt den `pagedown_bootstrap`-Stylesheet in die `application.css`-Datei.

```
markdown
/*
 * lib/assets
*= require flash
 */
```

Diese Anweisung lädt den `flash`-Stylesheet aus dem `lib/assets`-Verzeichnis in die `application.css`-Datei.

```
markdown
/*
 * app/assets
```

```
*= require_tree .
*/
```

Diese Anweisung lädt alle CSS- und SCSS-Dateien aus dem `app/assets`-Verzeichnis und seinen Unterverzeichnissen in die `application.css`-Datei. Die `.` steht dabei für das aktuelle Verzeichnis.

base.css.scss:

Dieser Code definiert CSS-Regeln für verschiedene HTML-Elemente und Klassen. Hier ist eine Zusammenfassung der Regeln:

- `h1` - definiert die Schriftgröße und den Abstand nach unten für Überschriften der Ebene 1
- `h1, h2, h3, h4, h5, h6` - definiert die Schriftart für alle Überschriften der Ebene 1-6
- `.lead` - definiert Schriftgröße und Farbe für Text, der als "Lead" gilt
- `a:not(.dropdown-item, .dropdown-toggle, .dropdown-link, .btn, .page-link), .btn-link` - definiert die Textdekoration für Links und Links mit Buttons, jedoch nicht für bestimmte Klassen
- `i.fa-solid, i.fa-regular, i.fa-solid` - definiert den Abstand auf der rechten Seite für verschiedene Font Awesome-Symbolklassen
- `pre, .output-element` - definiert die Hintergrundfarbe, den Abstand und den Rahmen für vorformatierten Text und ein benutzerdefiniertes HTML-Element
- `.btn` - definiert die Schriftglättung, das Gewicht und die Schriftgröße für Buttons, jedoch nicht für große Buttons
- `.btn-default` - definiert die Farbe des Randes für deaktivierte Buttons
- `.progress` - definiert den Abstand, den Rahmen und die Höhe für Fortschrittsbalken
- `.navbar` - definiert die Schriftglättung, das Gewicht und die Schriftgröße für die Navigationsleiste, sowie den Abstand für Dropdown-Items
- `.attribute-row + .attribute-row` - definiert den Abstand zwischen aufeinanderfolgenden Attributzeilen
- `.rounded-pill` - definiert die Schriftgröße, das Gewicht und den Abstand für abgerundete Elemente
- `.container[data-controller]` - definiert den unteren Abstand für einen Container, der ein bestimmtes Datenattribut verwendet
- `.disabled` - definiert den Cursor und die Opazität für deaktivierte Elemente
- `.empty` - definiert die Opazität für leere Elemente
- `.flash` - definiert die Schriftgröße und den Stil für einen Flash-Nachrichtenbereich mit Links
- `.markdown` - definiert die Höhe für einen Bereich, der Markdown-Text enthält
- `.spinner` - definiert die Größe, Hintergrundfarbe und Animation für einen Ladebildschirm
- `.table-row-clickable` - definiert den Cursor für anklickbare Tabellenzeilen
- `.inline_list` - definiert die Liste-Stilart, den Abstand auf der linken Seite und die Zeilenumbrüche für Inline-Listen
- `.single_entry_inline_list` - definiert die Liste-Stilart, den Abstand auf der linken Seite und die Zeilenumbrüche für einzelne Einträge in Inline-Listen

- `.modal-footer ul.horizontal` - definiert die Listen-Stilart und den Abstand für eine horizontale Liste in der Fußzeile eines Modal-Fensters, sowie den Trenner zwischen den Einträgen

bootstrap-dropdown-submenu.css.scss:

Dieser Code definiert das Styling für Dropdown-Menüs mit Untermenüs (Submenus) in CSS.

Die erste Regel (`.dropdown-submenu`) definiert die Positionierung der Untermenüs relativ zum Hauptmenü. Die nächste Regel (`.dropdown-submenu > .dropdown-menu`) definiert die Position des Untermenüs im Verhältnis zum Hauptmenü.

Die Regel `.dropdown-submenu.open > ul.dropdown-menu` stellt sicher, dass das Untermenü angezeigt wird, wenn das Hauptmenü geöffnet ist.

Die Regel `.dropdown-submenu > a:after` fügt eine nach rechts zeigende Pfeilspitze hinzu, um anzuzeigen, dass es ein Untermenü gibt. Die Pfeilfarbe ändert sich, wenn der Mauszeiger darüber schwebt.

Die Regel `.dropdown-submenu.float-start` definiert, dass das Hauptmenü links ausgerichtet ist und die Untermenüs rechts vom Hauptmenü angezeigt werden sollen. Der Abstand zwischen dem Hauptmenü und dem Untermenü wird durch `margin-left` festgelegt.

Die Border-Radius-Definitionen für die `.dropdown-submenu.float-start > .dropdown-menu` stellen sicher, dass die Ecken des Untermenüs abgerundet sind.

comments.css.scss:

Dieser Code definiert einen Stil für den Kommentarbereich eines Code-Editors namens Ace. Die Kommentare werden durch die CSS-Klasse `.ace_gutter-cell.code-ocean_comment` angesprochen.

Der Hintergrund des Kommentarbereichs wird durch die Eigenschaft `background-image` festgelegt und verwendet eine Base64-codierte PNG-Datei als Bildquelle. Die Eigenschaft `background-position` setzt das Hintergrundbild horizontal auf 2px und vertikal mittig.

editor.css.scss:

Dieser Code ist eine CSS-Datei und enthält verschiedene Stil-Definitionen für eine Webanwendung. Hier ist eine kurze Zusammenfassung der verschiedenen Abschnitte:

- **.editor:** Definiert die Höhe und Breite des Texteditors auf 100% der verfügbaren Größe.
- **.own-editor:** Definiert die Höhe und Breite des Texteditors auf 100% und legt auch den Hintergrund des Editors auf #FAFAFA fest.
- **.edit-frame:** Definiert die Höhe des Rahmens für die Übung, die bearbeitet wird, auf 400px. Legt auch das maximale Breitenverhältnis von Audio-, Bild- und Videoelementen auf 100% fest.
- **#content-left-sidebar, #content-right-sidebar:** Definiert eine Mindesthöhe von 250px für die linke und rechte Seitenleiste des Inhalts.
- **.frame und .own-frame:** Definiert eine Mindesthöhe von 300px für den Rahmen einer Übung und legt auch das maximale Breitenverhältnis von Audio-, Bild- und Videoelementen auf 100% fest. Der Unterschied zwischen .frame und .own-frame ist, dass .own-frame für den eigenen Rahmen des Benutzers verwendet wird.
- **.score:** Definiert die vertikale Ausrichtung der Bewertung unten.
- **#dummy:** Legt die Anzeige des Dummy-Elements auf "none" fest.
- **#editor-buttons:** Definiert den Hintergrund des Editor-Buttons auf #008CBA und gibt ihm eine Breite von 100%. Es verwendet auch das Flexbox-Layout für die Schaltflächen und Gruppen von Schaltflächen, um sie gleichmäßig auf den verfügbaren Platz zu verteilen.
- **#files und #own-files:** Definiert eine Scrollleiste für den Bereich der Dateien.
- **#output:** Legt den weißen Bereich fest, in dem die Ausgabe der Anwendung angezeigt wird. Es verwendet auch eine feste Breite von 14 Pixeln für die Schriftart und eine Scrollleiste für die Elemente der Ausgabe.
- **#results:** Legt die Anzeige des Ergebnisbereichs auf "none" fest.
- **#statusbar:** Legt die Sichtbarkeit der Statusleiste auf "hidden" fest und legt auch eine Höhe von 1,6em und eine Schriftfarbe von #777 fest.
- **#turtlecanvas:** Legt die Anzeige des Turtle-Canvas-Rahmens auf "block", gibt ihm eine Breite und eine Höhe und legt auch den Rahmenstil und die Breite fest.
- **.sidebar-col-collapsed und .sidebar-col:** Definieren die Breite der linken Seitenleiste auf 67px im zusammengeklappten Zustand und auf 20% im normalen Zustand. Sie verwenden auch das Flexbox-Layout für den Inhalt der Seitenleiste.
- **.editor-col:** Definiert die Mindesthöhe auf 1px, die Breite auf "auto" und die Höhe auf 100% und legt auch den Overflow auf "hidden" fest.
- **.output-col-collapsed und .output-col:** Definieren die Breite der rechten Ausgabespalte auf 67px im zusammengeklappten Zustand und auf 40% im normalen Zustand. Sie verwenden auch das Flexbox-Layout für den Inhalt der Ausgabespalte.

error_templates.scss:

Das CSS-Regelset `#add-attribute` definiert die Darstellung eines Formularfeldes zum Hinzufügen von Attributen. Es verwendet das Flexbox-Layout, um die Elemente horizontal anzuordnen. Die maximale Breite des Formularfeldes ist auf 400 Pixel begrenzt und es gibt einen Abstand von 30 Pixeln zum oberen Rand des umgebenden Elements.

Das darin enthaltene Regelset `button` legt den Abstand zwischen dem Button-Element und anderen Elementen im Formularfeld fest, indem es einen linken Rand von 10 Pixeln definiert.

exercise_collections.scss:

Dieser Code definiert verschiedene CSS-Stile für eine Visualisierung von Arbeitszeiten.

Die Arbeitszeit-Visualisierung besteht aus einem Linien- und einer Balkendiagramm. `path.line.minimum-working-time`, `path.line.average-working-time` und `path.line.maximum-working-time` definieren die Farbe der Linien im Linendiagramm, je nachdem, ob es sich um die minimale, durchschnittliche oder maximale Arbeitszeit handelt. Die Farben werden durch Verwendung der oben genannten Variablen `$min-color`, `$avg-color` und `$max-color` festgelegt.

Die Balken im Balkendiagramm werden durch die Klasse `rect.value-bar` definiert und haben die Hintergrundfarbe `$time-color`.

Die CSS-Regeln für den Legendenbereich sind in der Klasse `#legend` definiert. Die einzelnen Einträge in der Legende werden durch die Klasse `.legend-entry` dargestellt und haben jeweils eine Farb-Box, die je nach Art der Arbeitszeit (Mindest-, Durchschnitts-, Maximale oder Arbeitszeit insgesamt) farblich angepasst ist.

Die Klasse `.exercise-id-tooltip` definiert das Aussehen des Tooltips, der angezeigt wird, wenn der Mauszeiger über einem Element in der Visualisierung schwebt. Die Klasse `#exercise-list` definiert den Bereich, in dem die Arbeitszeiten angezeigt werden. Die Klasse `#add-exercise-list` definiert den Bereich, in dem neue Arbeitszeiten hinzugefügt werden können.

Schließlich werden durch die Klasse `.exercise-actions` die Schaltflächen definiert, die zur Steuerung der Arbeitszeit-Visualisierung verwendet werden. Die Schaltflächen haben einen Abstand von 10px zueinander.

exercises.css.scss:

Dies ist ein Cascading Style Sheet (CSS), welches für die Gestaltung von Webseiten verwendet wird. Hier werden verschiedene Stildefinitionen für HTML-Elemente angegeben.

Die erste Stildefinition legt die Hintergrundfarbe für ein Element mit der Klasse "exercise" fest und definiert eine maximale Höhe sowie Scroll-Verhalten für das Element.

Die zweite Stildefinition legt fest, dass ein Eingabeelement vom Typ "file" nicht angezeigt werden soll.

Die dritte Stildefinition gibt für das Element mit der ID "exercise_template_code" die Schriftart "monospace" vor.

Die vierte Stildefinition definiert verschiedene Stilregeln für Elemente mit der Klasse "exercise", wie zum Beispiel abgerundete Ecken, Schatten, Abstände und Positionierung.

Die nächsten beiden Stildefinitionen geben Anweisungen zur Darstellung von Achsen und Linien in einem Diagramm.

Weiter unten werden verschiedene Stile für Elemente mit bestimmten IDs definiert, zum Beispiel die Hintergrundfarbe für verschiedene Diagramme.

Die Stildefinitionen für die Klasse "feedback-page" geben verschiedene Regeln für die Gestaltung von Feedback-Elementen vor.

Die letzten Stildefinitionen legen das Aussehen von Elementen im Export-Modal fest, wie zum Beispiel die Breite und Höhe von Elementen und die Farbe für Erfolg oder Fehlermeldungen.

external_users.css.scss:

Die CSS-Selektoren beginnen mit "#" und beziehen sich auf ein HTML-Element mit einem bestimmten ID-Attribut.

- **"#no-elements"** versteckt das Element mit der ID "no-elements", indem es die CSS-Anzeige auf "none" setzt. Das Element wird also nicht auf der Webseite angezeigt.
- **"#tag-grid"** legt ein Gitterlayout für das Element mit der ID "tag-grid" fest. Das Gitter hat vier Spalten und jede Spalte ist 25% breit. Der Abstand zwischen den Spalten beträgt 10 Pixel und der Abstand zwischen den Zeilen beträgt 15 Pixel.
- **".progress"** bezieht sich auf alle Elemente mit der CSS-Klasse "progress" innerhalb des Elements mit der ID "tag-grid".
- **".progress-bar"** bezieht sich auf alle Elemente mit der CSS-Klasse "progress-bar", die innerhalb eines Elements mit der Klasse "progress" liegen. Es setzt die Mindestbreite der Fortschrittsbalken auf 2em.

file templates.css.scss:

Die Kommentare in diesem Code erklären, dass alle Stile, die sich auf den "FileTemplates"-Controller beziehen, in diesem CSS-File enthalten sein werden und automatisch in der Anwendung eingebunden werden. Zudem wird darauf hingewiesen, dass man Sass (SCSS) verwenden kann. Sass ist eine Erweiterung von CSS, die zusätzliche Funktionen wie Variablen, verschachtelte Regeln und Mixins bietet, um die Gestaltung von Webseiten zu vereinfachen.

flowrdata.css.scss:

Der Code definiert ein CSS-Stylesheet für das Element mit der ID "flowrHint". Dabei wird das Element standardmäßig ausgeblendet (mit "display: none") und es wird ein Abstand von 10 Pixeln zum oberen Elementrand (mit "margin-top: 10px") eingefügt.

forms.css.scss:

Dies ist CSS-Code, der auf verschiedene HTML-Elemente und Klassen angewendet werden kann. Hier sind die Bedeutungen der einzelnen Zeilen:

- `.alternative-input { display: none; }`: Alle Elemente, die die Klasse `.alternative-input` haben, werden ausgeblendet.
- `.chosen-container { min-width: 250px !important; width: 100% !important; }`: Alle Elemente, die die Klasse `.chosen-container` haben, haben eine Mindestbreite von 250 Pixeln und eine Breite von 100%. Das `!important` am Ende jeder Regel gibt dieser Priorität gegenüber anderen Regeln.
- `.chosen-inline { .chosen-container { min-width: unset !important; width: unset !important; } }`: Wenn die Klasse `.chosen-inline` auf ein Element angewendet wird, wird die Mindestbreite und Breite des darin enthaltenen Elements mit der Klasse `.chosen-container` auf ihre Standardwerte zurückgesetzt.
- `.code-field { font-family: monospace; }`: Alle Elemente, die die Klasse `.code-field` haben, verwenden die Schriftart "monospace".
- `.filter-form { .row:not(:last-child) { margin-bottom: 1em; } input, select { min-width: 200px !important; } }`: Alle Elemente, die die Klasse `.filter-form` haben, haben eine untergeordnete `div` mit der Klasse `.row`, deren Abstand zum unteren Rand 1em beträgt. Jede Eingabe- oder Auswahlbox innerhalb eines Elements mit der Klasse `.filter-form` hat eine Mindestbreite von 200 Pixeln.
- `.toggle-input { font-size: 80%; }`: Alle Elemente, die die Klasse `.toggle-input` haben, haben eine Schriftgröße von 80% im Vergleich zum Standardwert.

request-for-comments.css.scss:

Dieser Code definiert die Styles für verschiedene Elemente einer HTML-Seite. Die Styles werden über CSS-Klassen zugewiesen.

Die erste Klasse, `.rfc`, enthält Styles für verschiedene Abschnitte der Seite. Unter anderem gibt es Styles für Überschriften, Textblöcke, Zusammenklappen von Text, Knöpfe und Ergebnisse von Tests.

Die Klasse `#commentitor` ist für einen Kommentarbereich auf der Seite zuständig und definiert die Höhe und den Hintergrundfarbe.

Die Klasse `:not(.allow_ace_tooltip) > .ace_tooltip` sorgt dafür, dass bei einer bestimmten Bedingung ein Tooltip ausgeblendet wird.

Die Klasse `.modal-content` ist für ein Popup-Fenster zuständig und definiert die Mindesthöhe und -breite sowie die Flexbox-Eigenschaften für den Inhalt.

Die Klasse `.comment` definiert das Aussehen von Kommentaren, einschließlich Header, Inhalt, Editor und Aktionen.

Die Klasse `.comment-divider` definiert das Aussehen der horizontalen Trennlinie zwischen den Kommentaren.

Die Klasse `#otherComments` definiert die Ausrichtung von Kommentaren in einem Container-Element.

statistics.css.scss:

Der Code definiert verschiedene Stile, die auf HTML-Elemente angewendet werden können. Hier sind die Namen und Funktionen der verschiedenen Code-Blöcke:

1. `#submissions-slider`: Definiert die Marge am oberen und unteren Rand eines Elements mit der ID "submissions-slider".
2. `#current-file.editor`: Setzt die Höhe auf 400 Pixel des HTML-Elements mit der ID "current-file" und der Klasse "editor".
3. `.clickable`: Definiert den Cursor-Stil auf Zeiger, wenn der Mauszeiger auf ein Element mit der Klasse "clickable" zeigt.
4. `.flex-container` und `.flex-item`: Definieren eine flexible Box und ihre Eigenschaften. Die Klasse "flex-container" wird verwendet, um das Element als flexible Box zu definieren, während die Klasse "flex-item" verwendet wird, um ein Kind-Element innerhalb der flexiblen Box zu definieren und ihm ein Wachstumsverhältnis zu geben.
5. `#play-button`: Definiert die Höhe und Breite des HTML-Elements mit der ID "play-button" und gibt Marge an den oberen, unteren und rechten Seiten.
6. `div.unit-test-result`, `div.positive-result`, `div.unknown-result` und `div.negative-result`: Definieren das Aussehen von Kreisen, die verwendet werden können, um Testergebnisse anzuzeigen. Jeder Kreis hat eine andere Hintergrundfarbe und einen Schatten.
7. `tr.active`,
`tr:not(.before_deadline, .within_grace_period, .after_late_deadline)`,
und `tr.highlight`: Definieren die Hintergrundfarben und Filtereffekte für Tabellenzeilen in verschiedenen Zuständen. Die Klasse "active" wird verwendet, um die Hintergrundfarbe und Filtereffekte auf eine aktive Zeile anzuwenden, während "not" und andere Klassen verwendet werden, um sie auf Zeilen in anderen Zuständen anzuwenden.
8. `.before_deadline`, `.within_grace_period`, und `.after_late_deadline`: Definieren die Hintergrundfarben von Tabellenzeilen in verschiedenen Zeitfenstern.
9. `#statistics-container` und `.statistics-wrapper`: Definieren den Stil der Statistik-Kontainer. Die Klasse "statistics-wrapper" definiert eine Rasteranordnung, die aus vier gleich großen Spalten besteht. Jede Spalte enthält ein HTML-Element mit einem Link. Die Elemente haben einen blauen Rahmen, eine blaue Hintergrundfarbe und sind in der Mitte ausgerichtet. Innerhalb jedes Elements gibt es einen Titel und eine Datenanzeige. Die Klasse "group" wird verwendet, um eine Gruppierung der Elemente und ihre Formatierung zu definieren.