Dokumentation

Inhaltsverzeichnis

Installationsanleitung

Voraussetzungen

Node.js sollte installierte sein (getestet mit Version 18.12.1)

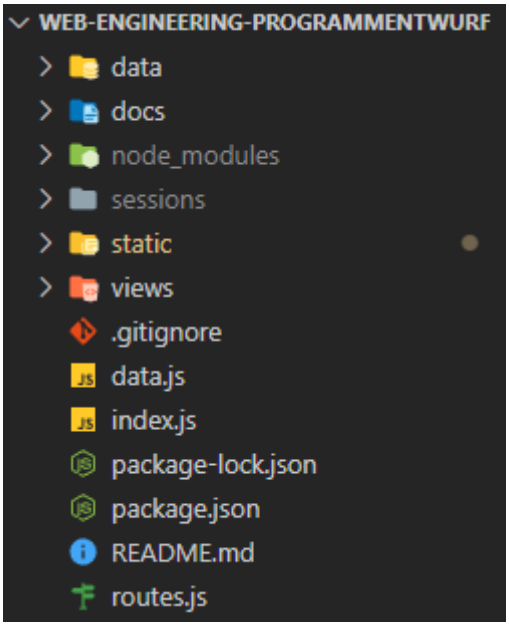
Installation

Folgende Befehle im Root-Verzeichnis ausführen:

1. npm install
2. npm run live

Verzeichnisstruktur

Auf root Ebene befinden sich folgende Ordner:



- Data: Darin befinden sich alle JSON-Dateien, die für die Speicherung von Daten benötigt werden.
- Sessions: Hier werden die Session Daten gespeichert.
- Static: Dort befinden sich alle statisch eingebundenen Dateien wie Bilder, clientseitige JavaScript- und CSS-Dateien.
- Views: Dort liegen alle ejs-Dateien, die von der View-Engine „EJS“ verwendet werden können

Alle anderen Dateien sind im root-Ordner selbst abgelegt.

package.json

In dieser Datei werden unterschiedliche Metadaten über eine Node.js App bzw. Projekt gespeichert. Dazu gehören beispielsweise der Name, die Version, eine Beschreibung und der Autor der App.

Außerdem wird hier mit dem keyword „main“ die Einstiegsdatei der App definiert. In unserem Fall ist dies die „index.js“, welche sich im root Verzeichnis befindet.

Im Block „scripts“ können Befehle, verbunden mit einem „key“, gespeichert werden. Zur leichteren Entwicklung verwenden wird beispielsweise den Befehl „nodemon index.js“, um die App lokal zu hosten. Das dazugehörige keyword ist „start“. Wenn man nun in der Konsole den Befehl „npm run start“ ausführt, wird der Befehl „nodemon index.js“ aufgerufen.

Ein weiterer wichtiger Block sind die „dependencies“. Hier sind alle npm-Pakete aufgelistet, die im Projekt verwendet werden. Mit einem einfach „npm install“ Befehl im Terminal, werden automatisch all diese Pakete installiert.

Index.js

Die index.js dient als Einstiegsdatei für das Node.js Projekt.

In den ersten acht Zeilen, werden alle benötigten Node-Module (npm-Pakete) eingebunden.

In Zeile 10 werden vier Funktionen eingebunden, die zur besseren Verwendbarkeit in eine externe Datei ausgelagert wurden.

In Zeile 12 wird der Port festgelegt, auf welchen die App lauschen soll.

In der der darauffolgenden Zeile wird eine neue Express-Anwendung erstellt. Diese wird dann in der Konstante „app“ gespeichert.
Express ist ein Web-Framework für Node.js. Es bietet eine Reihe von Funktionen, die das Erstellen von Webanwendungen und APIs erleichtern. Mit Express können unter anderem Routen definiert werden, um Anfragen an bestimmte URLs zu verarbeiten und Middleware verwendet werden, um Anfragen vor der Verarbeitung zu bearbeiten.

In Zeile 14 wird eine Instanz der „urlencoded“-Middleware erstellt und in der Konstante „encodeUrl“ gespeichert. Diese wird später in verschiedenen Routen verwendetet, um Anfragedaten im URL-codierten Format zu verarbeiten.

Weiterhin werden in Zeile 15 Optionen für den Session-Store gesetzt und in der Konstante „fileStoreOptions“ gespeichert. Dabei wird das Verzeichnis angegeben, in dem die Sessions gespeichert werden und die Anzahl, wie oft ein Speicherversuch durchgeführt werden soll.

In Zeile 21 werden alle vorhandenen Benutzer in die Variable „users“ gespeichert.

In Zeile 24 wird die „session“-Middleware zur Express Anwendung hinzugefügt. Hierbei werden verschiedene Option gesetzt. Wichtig zu erwähnen ist hierbei die „store“ und „genid“ Optionen. Als „store“ wird hier eine neue Instanz vom „FileStore“ zusammen mit der zuvor festgelegten Option (fileStoreOptions) erstellt.
Die Option „genid“ gibt eine Funktion an, die verwendet wird, um neue Sitzungs-IDs zu generieren. In unserem Fall wird die Funktion „uuid()“ verwendet, um UUIDs als Sitzungs-IDs zu verwenden.

In Zeile 41 wird ebenfalls eine Middleware hinzugefügt. Diese überprüft, ob in der aktuellen Session ein Benutzer gespeichert ist. Wenn ja, wird die „res.locals.authenticated“ Variable auf „true“ gesetzt, andernfalls auf „false“.

In Zeile 51 wird als Middleware das Favicon festgelegt.

Zeile 54 setzt die View Engine auf „ejs“.

Zeile 56 fügt einige Routen, die in einer externen Datei definiert sind, hinzu.

In Zeile 57 wird das Verzeichnis „/static“ statisch in die Anwendung eingebunden. Alle Dateien, die sich darin befinden, können dann durch Pfadangabe an den Client gesendet werden.

Routen

Im Folgenden werden die verwendeten Routen beschrieben.

Post: /auth

Hier werden zuerst der übergebene Benutzername und das Passwort aus dem Request-Body gespeichert. Anschließend wird versucht den Benutzer in der Variable „users“ zu finden.
Wird er gefunden, wird mit der Hilfe der „becrypt.compare“ Methode getestet, ob das eingegebene Passwort mit dem „gehashten“ Passwort im User-Objekt übereinstimmt. Wenn ja, wird eine Session erstellt und der Benutzer wird darin gespeichert. Danach wird der User zu Startseite geleitet. Wenn nein, bekommt der Benutzer eine Warnmeldung und wird zurück auf die Login-Seite gesendet.
Wird der Benutzer nicht gefunden, bekommt der Nutzer ebenfalls eine Meldung und wird zurück geleitet.

Post: /newUser

Zuerst werden hier alle übergebenen Eigenschaften aus dem Request-Body gespeichert. Danach wird geschaut, ob der Benutzername bereits existiert. Wenn ja wird dem Benutzer eine Meldung ausgegeben und er muss sich einen neuen Benutzernamen überlegen. Danach wird überprüft, ob das Passwort und die Wiederholte Eingabe übereinstimmen. Wenn nein, wird dem Benutzer ein Warnmeldung ausgegeben.
Wenn ja, kann der Benutzer nun gespeichert werden. Dazu wird zuerst mit Hilfe von „bcrypt“ das Passwort „gehasht“ und gesalzen. Anschließend werden alle vom Benutzer angegebenen Daten und das „gehashte“ Passwort in der Users-JSON gespeichert.

Post: /newComment

Zu Beginn werden alle Artikel aus der Artikel-JSON geladen. Danach wird die übergebene URL aus dem Request-Body gespeichert.
Nun wird überprüft, ob in der aktuellen Session ein Benutzerobjekt gespeichert ist. Wenn ja, wird anhand der übergebenen URL der Artikelname gespeichert. Anhand dieses Namens wird dann der richtige Artikel gesucht. In dieses Artikelobjekt werden dann der Benutzername, aktueller Zeitstempel und der Kommentarinhalt innerhalb des Kommentararrays gespeichert.
Ist in der aktuellen Session kein Benutzer gespeichert, bekommt der Nutzer den Hinweis, dass er sich anmelden muss.

Post: /logout

Wenn eine Session existiert, wird zuerst das Cookie des Benutzers gelöscht. Anschließen wird Session gelöscht und bei Erfolg wird der Nutzer zur Startseite geleitet.

Funktionen

generateTimestamp

Diese Funktion erstellt zuerst ein Datumsobjekt mit aktuellem Datum und Uhrzeit. In der Variablen „options“ werden verschiedene Eigenschaften zu Formatierung dieses Objekts gespeichert.
Mit der Methode „toLocaleTimeString“ wird das Datumsobjekt zusammen mit den Optionen dann in einem lesbaren deutschen Format ausgegeben.

Zum Schluss wird in der index.js die „app.listen()-Methode“ aufgerufen. Damit wird der Webserver auf dem zuvor festgelegten Port gestartet.

Kommentarfunktion

HTML (comments.ejs)

Die Kommentarsektion besteht aus zwei großen Bereichen: Ein Verfassungs- und ein Anzeigebereich.

Der Verfassungsbereich besteht aus einer HTML-Form. Diese Form sendet beim Absenden einen Post-Request an die Route „/newComment“. Die Form beinhaltet eine „Textarea“, in der der Benutzer seinen Kommentar verfassen kann. Das Textfeld hat eine maximale Zeichenlänge von 1000 und eine minimale Zeichenlänge von 10.
Außerdem gibt es ein unsichtbaren „Inputfield“ mit dem Namen „url“. Hier wird über clientseitiges JavaScript die aktuelle URL gespeichert.
In einem Container unterhalb des Textfeldes wird außerdem angezeigt, wie viele Zeichen man bereits eingegeben hat.
Am Ende befindet sich ein „submit-Button“, mit dem der Kommentar abgesendet werden kann.

Der Anzeigebereich besteht aus einem Informationsblock, der den Kommentarverfasser, einen Zeitstempel anzeigt und den Kommentar an sich.
Der Anzeigebereich wird dynamisch mit Hilfe von „ejs“ generiert und erweitert sich je nach Anzahl der Kommentare. Mit „ejs“ wird zuerst geprüft, ob das Kommentarobjekt vorhanden ist. Wenn ja, wird zusätzlich geprüft ob darin Kommentare vorhanden sind. Trifft auch das zu wird der Anzeigebereich so oft ausgegeben, wie es Kommentare gibt. Zusätzlich wird auch der Inhalt dieser Kommentare (Benutzername, Zeitstempel und Kommentarinhalt) dynamisch aus dem Kommentarobjekt ausgelesen und ausgegeben.

CSS (comments.css)

Folgend werden nur komplexere Style-Eigenschaften beschrieben. Eigenschaften wie „padding“ oder „margin“ sind in mehreren Elementen definiert und bestimmen dabei den Innen- bzw. Außenabstand zwischen den einzelnen Elementen. Auch die Eigenschaften „height“ und „width“ bestimmen lediglich die Höhe und die Breite der Elemente und haben sonst keine tiefere Bedeutung.

Comments-section-wrapper:

Dieser Style umfasst die ganze Kommentarsektion und setzt deren Breite auf 100%. Display: flex bewirkt, dass sich alle untergeordneten Elemente flexibel anordnen. Mit flex-wrap: wrap wird festgelegt, dass sich die Elemente bei Bedarf umgebrochen werden.
Bei fast allen untergeordneten Elementen die Breite auf 100% gesetzt, damit eine Vertikale Anordnung entsteht.

Comment-write-wrapper:

Dieser Style umfasst den Verfassungsbereich. Eine wichtige Eigenschaft ist hier „border-bottom“. Damit wird unterhalb des Containers eine Linie erzeugt, die eine sichtbare Abgrenzung zwischen Verfassungs- und Anzeigebereich darstellt.

Comment-form:

Bezieht sich auf die HTML-Form. Hier wird die Eigenschaft text-align: left gesetzt, damit alle sich darin befindlichen Textelemente linksbündig angezeigt werden.

CommentInput:

Beschreibt das Aussehen der HTML-textarea. Mit „resize: none“ wird festgelegt, dass die Größe des Textfelds nicht durch den Benutzer verändert werden kann. Außerdem wird mit „font-family“ die Schriftart gesetzt. Mit „border-radius“ werden an den Ecken des Textfelds kleine Rundungen hinzugefügt.
Ist das Textfeld im Fokus (commentInput:focus) wird die „border“ auf 2px vergrößert, um dem Nutzer visuell zu zeigen, dass er gerade das Textfeld benutzt. Die Eigenschaft „outline“ wird auf „none“ gesetzt, da sonst noch eine zusätzliche „border“ angezeigt wird.

Comment-button:

Beschreibt das Aussehen des Buttons zum Absenden des Kommentars. Mit „color: black“ wird die Schriftfarbe auf Schwarz gesetzt. Mit der Eigenschaft „transition: 0.3s“ wird die Dauer von Übergängen festgelegt. Dieser kommt zum Einsatz, wenn der Benutzer seine Maus über den Button bewegt (comment-button:hover). Dann wird die Hintergrundfarbe des Buttons etwas dunkler gemacht. Durch den „transition-Effekt“ dauert diese Veränderung 0.3 Sekunden.

Comments-wrapper:

Umfasst die Styles für den Anzeigebereich. Mit „display: flex“ und „flex-wrap: wrap“ wird hier wieder eine flexible Anordnung der Unterelemente bewirkt, die bei bedarf automatisch einen Umbruch auslösen. Bei fast allen untergeordneten Elementen die Breite auf 100% gesetzt, damit eine Vertikale Anordnung entsteht.

Comment-info:

Beinhaltet den Benutzernamen und den Zeitstempel. Mit „display: flex“ und „justify-content: space-between“ werden diese beiden Elemente jeweils ganz links und ganz rechts platziert.

Comment:

Beinhaltet den Kommentarinhalt. Hier wird zur besseren Abgrenzung eine „border“ von 1px gesetzt.

JS (comments.js)

Zu Beginn werden das Kommentarfeld (commentInput) und die Anzeige für die Anzahl der Zeichen (character-count) jeweils in eine Konstante gespeichert.

Dann wird ein „Event-Listener“ zu dem Kommentarfeld hinzufügt. Der „Event-Listener“ wird jedes Mal ausgelöst, wenn der Benutzer etwas in das Kommentarfeld eingibt ("input"-Event). Wenn das Event ausgelöst wird, wird eine Funktion ausgeführt, die das Event-Objekt als Argument erhält. Innerhalb dieser Funktion wird das Ziel des Events (e.target) in einer Variablen gespeichert.
Dann wird das „maxlength-Attribut“ des Ziel-Elements abgerufen und in einer Variablen namens „maxLength“ gespeichert. Anschließend wird die aktuelle Anzahl der Zeichen im Ziel-Element gezählt und in einer Variablen namens „currentLength“ gespeichert.
Zum Schluss wird der Inhalt des Elements „character-count“ aktualisiert, um die aktuelle Anzahl der Zeichen und die maximale Anzahl der Zeichen anzuzeigen (z.B. “25/1000”).

Die letzte Code-Zeile speichert die aktuelle URL als „value“ im „url“-Input der HTML-Form.

Datenspeicherung

Alle persistenten Daten werden in JSON-Dateien gespeichert. Diese JSON-Dateien liegen im Verzeichnis „./data“.

Articles.json

Im Objekt „articles“ werden hier die Namen aller Artikel inklusiver ihrer Kommentare gespeichert. Der Name wird im Wert des Schlüssels „articleName“ gespeichert. Der Wert des Schlüssels "comments" ist ein Array von Objekten, wobei jedes Objekt einen Kommentar darstellt. Jeder Kommentar hat dabei die Schlüssel „username“ für den Benutzernamen, „timestamp“ für den Zeitstempel und „comment“ für den Inhalt des Kommentars.

Users.json

Im Objekt „users“ werde hier alle Benutzer gespeichert. Als Werte werden hier pro Benutzer der Benutzername, Vorname, Nachname, das Passwort als Hash und das Geschlecht gespeichert.

Registrierung

HTML (registration.ejs)

Die HTML besteht hauptsächlich aus einer Form, die von einem Container umgeben wird.

Die Form beginnt mit einer Überschrift (h2). Anschließen folgen mehrere Input-Container mit der Klasse „input-wrapper“, die sich alle ähnlich sind. Sie bestehen aus einem Label und einem nachfolgenden Input.
Alle input-Felder haben das Attribut „required“, einen Platzhaltertext und einen entsprechenden Namen.
Die Inputs für den Benutzernamen, Vornamen und Nachnamen haben zusätzlich ein Attribut

„maxLenght=20“, um die maximale Zeichenanzahl auf 20 zu beschränken. Außerdem wird mit dem Attribut „pattern“ mit Hilfe von regulären Ausdrücken beschrieben, dass nur bestimmte Zeichen (z.B. a-Z) zulässig sind.

Für die Wahl des Geschlechts wurden Radio-Buttons verwendet. Da beide Buttons den gleichen Namen haben, sind diese miteinander verbunden. Wir also ein Button ausgewählt, wird der andere automatisch abgewählt.

Für das Akzeptieren der Nutzungsbedingung wird eine einfach Checkbox verwendet.

Zum Schluss kommt ein Button vom Typ „submit“, um die HTML-Form abzusenden. Dabei wird ein POST an die Route „/newUser“ mit allen eingetragenen Daten gesendet.

CSS (registration.css)

Beim „body“ wird zuerst die Schriftart gesetzt. Mit „display: flex“ kann das untergeordnete Element flexibel platziert werden. Die „height“ wird auf 100vh gesetzt, um die volle Bildschirmhöhe einzunehmen. Zusätzlich wird das „margin“ auf 0px gesetzt, um einen Overflow zu vermeiden. Als Hintergrund wird ein Bild verwendet. Dieses soll nicht wiederholt werden (no-repeat) und den gesamten Platz einnehmen (cover).

Registration-wrapper:

Um die Breite festzulegen wird hier „max-width (550px)“ statt „width“ benutzt. Das bewirkt, dass, bei geringerer Breite vom Browser, der Container automatisch kleiner wird. Mit „margin: auto“ platziert sich der Container horizontal mittig. Die Hintergrundfarbe wird mit „rgba(255, 255, 255, 0.6)“ auf Weiß mit einer Transparenz von 0.6 festgelegt. Der border-radius rundet die Ecken ein wenig ab und durch das „padding“ von 8px haben alle sich darin befindlichen Elemente einen Abstand zum Rand.

h2:

Mit einer Breite von 100% und „text-align: center“ wird die Überschrift horizontal mittig platziert.

Input-block:

Mit „display: flex“ und „flex: wrap“ werden die untergeordneten Elemente flexibel angeordnet und bei Bedarf umgebrochen. Die Breite ist auf 100% gesetzt und die minimale Höhe beträgt 50px.
„Justify-content: space-between“ wird dann benötigt, wenn zwei Input-Elemente nebeneinander dargestellt werden sollen.

Double-column:
Durch flex-basis: 48% wird die Breite dieser Container festgelegt. Da wie zuvor erwähnt im Elternelement die „justify-content“ Eigenschaft gesetzt ist, werden die Container link und rechts mit Platz dazwischen angeordnet.

Input:

Mit „width: 100%“ wird immer die maximale Breite genutzt. Margin-top: 4px sorgt für einen Abstand zwischen Input und darüberliegenden Label. Die standardmäßige Border wird durch eine neue mit 1px Breite ersetzt. Um alle Abstände richtig zu berechnen wird „box-sizing“ auf „border-box“ festgelegt.
Margin-Bottom sorgt für einen Abstand zum nächsten input-block.

Block-header:

Wird als Überschrift für das Geschlecht benötigt und bekommt eine Breite von 100% und wird fett gedruckt.

Label:

Alle Label bekommen eine Breite von 100%, um einen Umbruch auszulösen (flex-wrap).

Input[type=text & type=password]:

Border-Radius für das Abrunden der Ecken und Padding, damit der eingegebene Text einen kleinen Abstand zur Border hat.

Input[type=radio & type=checkbox]:

Margin um einen Abstand zu den angrenzenden Elementen zu generieren. „Width: auto“, damit die Inputs auf einer Höhe mit dem Label sind.

Agb-wrapper:

Alle Eigenschaften von diesem Container sorgen dafür, dass die untergeordneten Elemente flexibel, rückwärts (row-reverse) und beginnend am Ende (flex-end) platziert werden. Das darin befindliche Label wird mit „width: auto“ auf dieselbe Höhe wie die „checkbox“ gesetzt.

Button-wrapper:

Mit „align-items: center“ wird der Button vertikal mittig platziert.

Button:

Breite wird auf 100% gesetzt und die Höhe auf 40px. Die Border wird entfernt und die Schrift wird auf 1em und fett gesetzt. Hinzu kommt eine Transition von 0.3s, die dann einen Sinn bekommt, wenn wir uns die Hover-Eigenschaft genauer anschauen.
Hier wird die background-color auf Grau gesetzt. Durch die Transition dauert diese Veränderung nun 0.3. Außerdem wird der Cursor zum Pointer verändert.

Artikel

Besteht aus einem Globalen Head und „Footer“ welche in extra Dateien liegen zur schnellen Wiederverwendung. Nach dem Head folgt ein Container welche den Artikel beinhaltet, der Artikel beginnt mit einer passenden Bild gefolgt mit einem Text zu dem jeweiligen Thema. Unterhalb des Artikels befindet sich noch eine Weiterleitung zu weiteren Artikeln mit demselben Thema.

Article.css

Besteht aus den Styles für die Artikel.

Body-Style:

Setzt die allgemeine Body Farbe auf den Hex #f2f2f2 ein hellgrau.

Span1:

Die Klasse span1 stylt Unterüberschriften innerhalb eines Artikels, es wird die Größe auf „1rem“ und der Zeilenabstand auf 1.7 gesetzt um es vom regulärem Text hervor zu heben.

Container-Style:

Der „.container“-Style enthält verschiedene Regeln, um den Container auf der Seite zu stylen, der den Inhalt der Website enthält. Es wird „flex“ als „display“ verwendet, um die Elemente im Container anzuordnen. Die Ausrichtung erfolgt in Spaltenrichtung und die Ausrichtung erfolgt an der linken Seite. Die Hintergrundfarbe wird auf Weiß gesetzt und es wird ein Abstand von 40 Pixeln um den Container hinzugefügt. Die maximale Breite des Containers beträgt 1200 Pixel, und der Container ist horizontal zentriert. Der Container wird auch eine z-index-Eigenschaft haben, um zu ermöglichen, dass andere Elemente auf der Seite unter oder über dem Container angezeigt werden können. Außerdem wird dem Container eine abgerundete Ecken-Eigenschaft mit einem Radius von 5 Pixeln zugewiesen.

Container-Image-Style:

Dieser Style wendet sich an das Bild innerhalb des Containers. Das Bild wird auf eine Breite von 100% eingestellt und der Höhe wird ein Wert von „auto“ zugewiesen, damit es sich automatisch an die Breite des Containers anpasst. Es wird auch ein Abstand von 10 Pixeln am unteren Rand des Bildes hinzugefügt, um es vom Text abzugrenzen. Das Bild wird auch abgerundete Ecken mit einem Radius von 5 Pixeln haben.

Container-Text-Style:

Dieser Style wird auf alle Textelemente innerhalb des Containers angewendet, die nicht Überschriften oder Bilder sind. Der Text wird in einer Schriftgröße von 0,8rem angezeigt, und der Zeilenabstand beträgt 1,3.

Small-images-Style:

Dieser Style wird auf die Bilder am Ende des Artikels angewendet. Die Bilder werden in einer horizontalen Reihe mit einer zentralen Ausrichtung und einem Abstand von 10 Pixeln zwischen ihnen angezeigt. Sie haben eine Breite von 300 Pixeln und eine Höhe von "auto", um sich automatisch an die Breite des Containers anzupassen. Sie haben auch abgerundete Ecken mit einem Radius von 5 Pixeln und eine schwarze Umrandung. Wenn Sie auf eines der Bilder zeigen, wird es 10% größer skaliert. Die Bilder haben auch einen Cursor in Form einer Hand, um anzuzeigen, dass sie anklickbar sind.

Small-images-Links-Style:

Dieser Style wird auf die Links angewendet, die sich um jedes der kleinen Bilder befinden. Die Links werden in einer Spalte angeordnet und die Ausrichtung erfolgt an der linken Seite. Die Breite des Links beträgt 200 Pixel, mit einem Abstand von 50px jeweils links und rechts.

Small-images-a-Img-Style:

Mit „object-fit: cover“ wird das Bild so skaliert, dass es vollständig in den Container passt, one Seitenverhältnisse zu verzerren. Die Höhe wird auf 100% gesetzt und ein Rechter Abstand von 10px zwischen den Bildcontainern wird gesetzt.

Small-images-a-text-Style:

Setzt den oberen Abstand zu den Bildern auf 20px und den untern und seitlichen auf 0.

Small-images-text-Style:

Schriftgröße wird auf 0.8rem festgelegt, innerhalb des Containers wird der Text zentriert und der Abstand nach oben wird auf 10px gesetzt.

Medienabfrage:

Medienabfrage bei Bildschirmen mit maximaler Breite von 768px.

Die Flex-Ausrichtung wird in eine Spalte geändert, um auf kleinere Bildschirme zu passen, und die Ausrichtung wird linksbündig ausgerichtet. Der obere Abstand des Bildes wird auf 20px geändert und das Bild wird so skaliert, dass es vollständig in den Container passt. Der obere und untere Abstand wird auf 40px festgelegt. Der Abstand zwischen den Bildcontainern wird auf 20px festgelegt. Der Abstand für jedes dritte Bild (beginnend mit dem ersten Bild) wird auf 0px gesetzt. Der Abstand für jedes dritte Bild (beginnend mit dem letzten Bild) wird auf 0px gesetzt.

Login

Beinhaltet Globalen Head und „Footer“, ebenso hat es das Anmelde Formular im Zentrum. Dieses benötigt den Benutzernamen und das Passwort mit welche dann als Formular abgesendet werden zur Weiterverarbeitung. Auch hat es hier einen Knopf welcher einen zum Registrierungsformular weiterleitet. Der Hintergrund ist ein Slider welchen zwischen drei Bildern rotiert.

Login-Stylesheet.css

@keyframes slide:

Definiert die Schritte in der sich die „slides“ von links nach rechts bewegen, dies wird mit der Eigenschaft „translateX“ angewandt. Mit den „animation-timing-function“ wird der Übergang der Bewegung bestimmt, für einen flüssigen Übergang der Bilder.

Slider:

Hält den Inhalt innerhalb des sichtbaren Bereichs mit der Eigenschaft „hidden“ für den „overflow“. Der Inhalt wird auf die volle breite des Bildschirms gestreckt und die Höhe wird auf 90% begrenzt.

Slider-slide:

Definiert die Position der einzelnen slides damit diese oberhalb des „elternelements“ zentriert bleiben und sich über den ganzen Bildschirm erstrecken. Schließlich wird die "slide"-Animation mit der Eigenschaft „animation: slide 16.5s infinite“ angewendet, um das Element in einer Dauerschleife zu animieren.

Slider-slide:nth-child:

Definiert die einzelnen Bilder der jeweiligen slides und teilt die Zeit auf die drei slides auf damit diese gleich lange sichtbar sind.

Login-container:

Positioniert den Container für den Login Horizontal zentriert, mit einer Breite von 400px und einer Höhe von 480px. Es hat einen box Schatten damit dieser einen leichten 3D Effekt aufweist, mit leicht abgerundetem Rand. Die Farbe des Containers wird auf ein leicht transparentes Weiß gesetzt.

Labels:

Die einzelnen labels werden untereinander angeordnet mit deren jeweiligen „inputs“ direkt unter sich.

Button:

Der „submit- und register-button“ sind im Container ganz unten angeordnet und sind auch transparent mit einer „hover“ Eigenschaft welches sie nichtmehr transparent lässt.

Show-password:

Positioniert die „checkbox“ neben das dazu gehörige Label.

Medienabfrage:

Medienabfrage bei Bildschirmen mit maximaler Breite von 768px.

Lässt den Slider automatisch kleiner machen bei kleineren Bildschirmen.

Login-js.js:

function togglePassword():

Ermöglicht es das Passwort als sichtbaren text anzuzeigen.

Dropdownmenü

Die Navigationsleiste aller HTML-Seiten ist im Stil eines Dropdownmenüs erstellt. Im Folgenden wird beschrieben, wie wir diese Navigationsleiste erstellt und gestylte haben.

- Mit „:root“ wer die Variablen für die Hintergrundfarben und die Höhe des Dropdownmenüs festgelegt.
- Mit „*“ werden Styles festgelegt, die für alle HTML-Elemente gelten.
- Mit „nav“ werden die Höhe und die Farbe der Navigationsleiste festgelegt. Diese wird vertikal zentriert.
- Mit „nav .logo“ wird das Logo der Seite gestylte und dessen Position festgelegt.
- Mit „nav ul“ wird die eigentliche Navigation festgelegt. Sie wird ans Ende der Zeile gesetzt und die einzelnen Menüpunkte werden in einer Reihe angeordnet. Die Stichpunkte der Menüpunkte werden entfernt.
- Mit „nav li“ wird die Größe der Kästen der einzelnen Menüpunkte festgelegt.
- Mit „nav li:hover“ ändern die Kästen ihre Farbe, wenn der Mauszeiger innerhalb dieser ist, zu der Farbe, die im „root“ als „—accent-color“ deklariert wurde.
- Mit „nav ul a“ wird dafür gesorgt, dass der Link schon am unteren Rand des Kästchens anklickbar ist und nicht erst wenn der Mauszeiger direkt über dem Link ist. Der Link nimmt also die gesamte Größe des „li“ ein. Zudem werden die Unterstreichungen der Links mit „text-decoration:none“ entfernt.
- Mit „.dropdown“ wird das eigentliche dropdown erstellt. Es ist wichtig, dass hier und im „li“ die Position „position:relative“ ist, da das Dropdown sich relativ zu seinem „li“ positionieren soll. Das Dropdown wird unterhalb der Navigationsleiste positioniert, hierfür gibt es im „root“ die Variable „—navbar-height“
- Mit „dropdown li“ werden die Kästen der Menüpunkte im Dropdown gestylte. Sie sind so breit wie die Menüpunkte der Navigationsleiste.
- Mit „dropdown li a“ wird der Text der Menüpunkte linksbündig. Mit „padding“ wird der Abstand zum linken Bildschirmrand bestimmt. Aufgrund des „padding“, muss die Breite angepasst werden, da die Punkte sonst schon außerhalb der Box angeklickt werden können.

- Mit „nav li:hover .droopdown“ wird das Dropdownmenü erst dann sichtbar, wenn man mit dem Mauszeiger über den Menüpunkt fährt, bei welchem das Dropdownmenü sichtbar werden soll.
- Mit „nav input[type=“checkbox“]“ wird die checkbox vor dem Menüpunkt mit dem Dropdownfunktion entfernt.
- Mit „.expandable_li“ wird der Text des Menüpunkts mit Dropdownfunktion so formatiert, wie der Text der anderen Kästen.
- Mit „@media“ wird die Seite „responsive“ gemacht.

Quiz

quiz.css

In der css-Datei „quiz.css“ werden die Checkboxen des Quiz erstellt und gestylte. Wir sind folgendermaßen vorgegangen…

- Mit „h1“ und „span“ wird die Überschrift der „quiz.ejs“ Seite so formatiert und gestylte, dass sie im Stil des Webseitenlogos ist.
- Mit „.inputcontainer“ werden die „container“ der einzelnen Fragen zu einer Klasse zusammengefasst. Jede Frage hat daher einen „.inputcontainer“.
- Mit „.inputcontainer input“ wird die vom Browser automatisch erzeugte checkbox entfernt.
- Mit „.checkmark“ wird wird eine custom checkbox erstellt.
- Mit „.inputcontainer:hover input~.checkmark“ werden die checkboxen grau, wenn man mit dem Mauszeiger darüber fährt.
- Mit „input[type=checkbox]:checked+.checkmark“ wird die checkbox beim Klicken auf diese Grau hinterlegt.
- Mit „.inputcontainer input:checked~.checkmark:after“ bleibt die Box nach dem Anklicken grau hinterlegt.
- Mit „.inputcontainer .checkmark:after“ wird die checkmark so gestylte, wie sie am Ende nach dem Anklicken der Box zu sehen ist.
- Mit „#change-color-button“ wird der button, der zum Überprüfen der markierten Kästchen geklickt wird mit einem Rahmen versehen und dessen Größe und Farbe wird festgelegt.
- Mit „#change-color-button:hover“ nimmt der Button die Akzentfarbe an und die Textfarbe wird weiß, wenn der Mauszeiger über dem Button ist.
- Mit „.container“ wird die weiße „box“ erstellt, in der sich das Quiz befindet.

Quiz (quiz.js)

Im JavaScript „quiz.js“ wird zunächst die Variable „changeColorButton“ deklariert. Mit „querySelector“ werden anhand der angegebenen „id’s“ Elemente im HTML-File ausgewählt. Diese werden dann für die Erstellung des Quiz in Variablen gespeichert.
Somit benötigt jede Antwortmöglichkeit eine eigene „id“ und eine eigene Variable im JavaScript mit der zugehörigen „id“.

Allgemeines Codebeispiel:

var=Variablenname

const var = document.querySelector(,#id‘);

if (var.checked) {

  var.nextElementSibling.style.backgroundColor = 'red'/‘green‘;

} else {

  var.nextElementSibling.style.backgroundColor = '';

}

Der obenstehende Code unterscheidet sich in der „id“ und im Variablennamen. Jede Antwortmöglichkeit hat seine eigene „id“ und seine eigene Variable.
Ist die Antwort falsch, so wird die angeklickte Box rot, ist sie richtig, wird sie grün. Hierfür wird die „if/else“ Funktion verwendet. Die Farben ändern sich erst, wenn der Button gedrückt wird. Es können mehrere Felder ausgewählt werden, da es mehr als eine richtige Antwort geben kann. Die „checkboxen“ ändern ihre Farbe erst beim Drücken des „submit-Buttons“, aufgrund des „eventListeners“, welcher die „function()“, erst startet, wenn man auf den Button klickt.

HTML-File der einzelnen Artikel

Die Artikel, die in den Ordnern „skifahren“, „bodybuiling“ und „klettern“ haben identische Stylesheets, um die Webseite einheitlich zu halten. Sie unterscheiden sich nur im Inhalt der Artikel und den verwendeten Bildern. Jeder Artikel hat ein eigenes Kommentarfeld, um individuell zu jedem Artikel Kommentare verfassen zu können. Diese werden am unteren Rand des Artikels angezeigt und sind für jeden sichtbar.

Partials

Footer

Beinhaltet den Kontakt zu den drei Projekt Teilnehmern und ist auf allen Seiten global als „Footer“ eingestellt. Befindet sich in einer externen Datei für schnellere und leichteres einfügen.

partialStyle.css

Footer-Style:

Hat die eine „linear gradiente“-farbe, und streckt sich auf die ganze Breite des Bildschirmes

Footer-Kontakt-Style:

Ordnet die Kontakte in einen „grid“ an um die Position der Namen und E-Mail-Adressen zu vereinfachen.

Kontakt:

Text wird jeweils zentriert innerhalb der „grid area“ und hat eine weiße Farbe mit einer Größe von 12px.

Startseite

Beinhaltet kleinen „Wilkommenstext“ zu der Website, unterhalb des Textes befinden sich drei Weiterleitungen zu drei aktuell beliebten Artikeln.

Startseite.css

Body:

Farbe wird auf ein hellgrau gesetzt mit dem hex #f2f2f2.

Span:

Farbe wird auf rot gesetzt.

Button:

Button Anordnung wird gesetzt mit transparentem Hintergrund und einem pointer als cursor.

Buttoncontainer:

Die Größe des Containers für die Buttons wird auf eine Breite von 200px gesetzt und der z-index auf 2 damit es immer oberhalb liegt.

Insta/Twitter/Facebook Button:

Positioniert die einzelnen Buttons. Sind untereinander angeordnet und haben die „hover-Eingenschaft“ versteckt, ebenso haben sie eine „hover-animation“ welche sie 5px nach rechts verschiebt.

Container:

Container wird zentriert und schmaler als der Body definiert damit links und rechts Balken mit Farbe des Bodys entstehen. Der z-index wird auf 1 gesetzt damit dieser immer oberhalb des Bodys liegt. Die ecken werden mit einem „border-radius“ von 5px abgerundet.

Artikel:

Artikel wird innerhalb des Containers zentriert und erhält einen Abstand nach links und rechts. Die Schriftgröße der Überschrift h1 wird auf groß gesetzt mit einem Zeilenabstand von 2, damit diese sich von Rest abheben kann.

Images:

Diese werden unterhalb des Textes innerhalb des Containers positioniert. Sie sind in einer Reihe angeordnet zu einander angeordnet, zu dem Text unter sich sind sie jeweils in einer Spalte angeordnet. Sie haben auch eine „hover animation“ welche sie mit zu 1.1 skaliert.

Medienabfrage:

Medienabfrage bei Bildschirmen mit maximaler Breite von 768px.

Sorgt für eine automatische Anpassung bei kleineren Bildschirmen. Mit „padding“ und „margin“ wird für eine passende Anpassung gesorgt.

Routes

Hier wird zuerst das „express framework“ definiert und auch das darauf basierend die Route-Definition, ebenso wird definiert das man das „data-Verzeichnis“ benötigt um auf Inhalte wie Nutzer und Kommentare zugreifen zu können. Im nächsten Teil werden die ganzen Routen definiert, wird eine HTML-GET-Request am die jeweilige URL gesendet wird diese gerendert. In den oberen drei Routen wird noch die Methode „getComments“ aufgerufen welche aus dem „data-Verzeichnis“ die zu dem jeweiligen Artikel gehörenden Kommentare zurückgibt. Beim Rändern der Seite werden hier dann auch noch die Kommentare mit übergeben, welche dann auf der Seite auch zu sehen werden. Die restlichen Routen werden einfach ohne die Kommentare gerendert. Zum Schluss wird das Modul noch exportiert, damit auch die „index.js“ Zugriff auf die einzelnen „Routes“ bekommt.

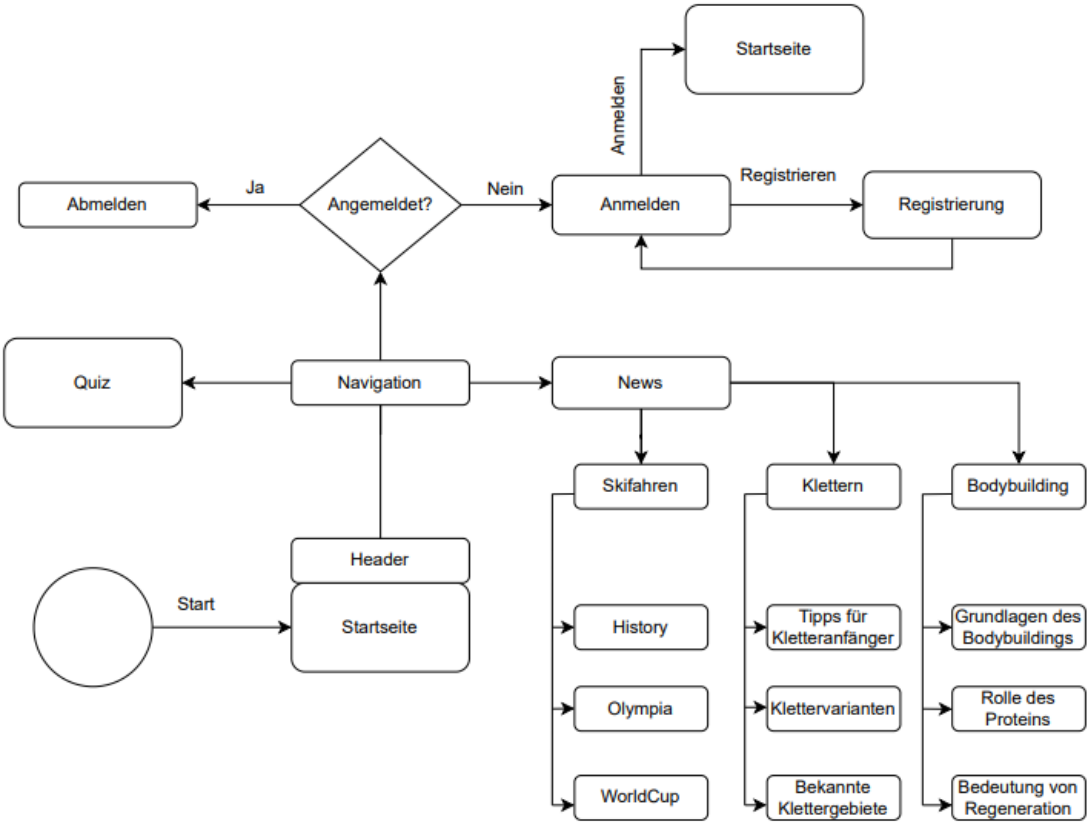
Ejs

"EJS" steht für "Embedded JavaScript", was eine Vorlage (Template) Sprache für JavaScript ist. Es wird oft verwendet, um dynamische Webseiten zu erstellen, indem es erlaubt, dass HTML-Code mit JavaScript-Code vermischt wird.

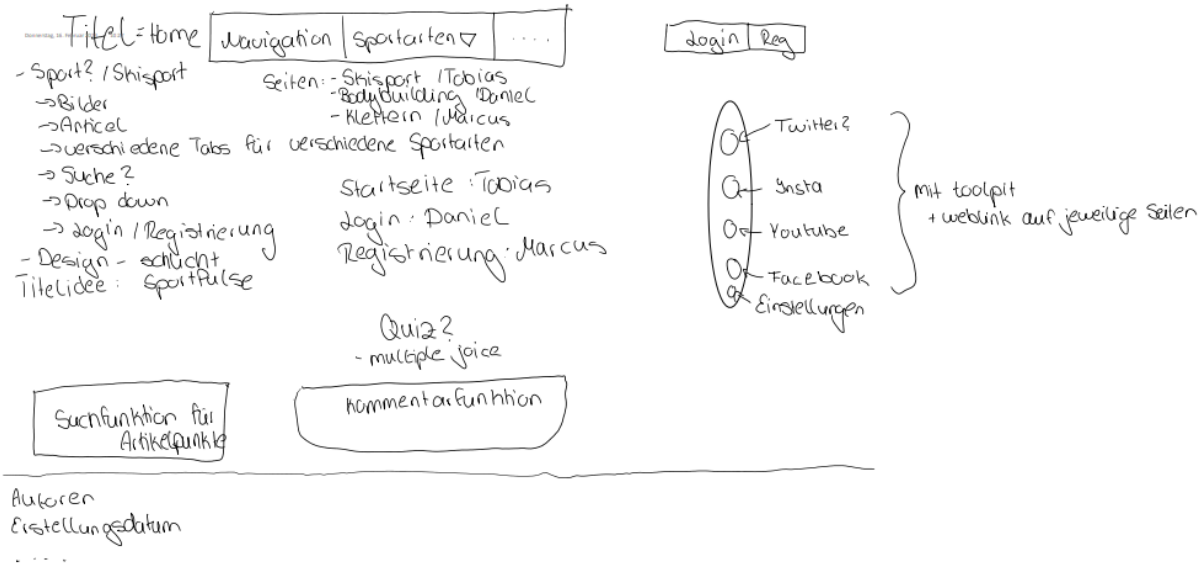
Wir haben für unsere Seite „EJS“ verwendet, da es den code deutlich übersichtlicher und dadurch auch besser lesbar macht. Zusätzlich sind footer, head auf jeder Seite gleich und die Kommentare sind ebenfalls auf jeder Artikelseite. Durch die Verwendung von „EJS“ haben wir also die häufige Wiederholung von HTML-Code verhindert.

Anlagen

Programmablaufplan



Grundidee



Quellen

- DigitalOcean | The Cloud for Builders
- diagrams.net
- Express - Node.js web application framework (expressjs.com)
- W3Schools Online Web Tutorials