



ENTWICKLERDOKUMENTATION

Inhaltsverzeichnis

Technische Anforderungen:.....	3
Aufteilung des Projektes:.....	3
Gauss_Calculator:	3
Gauss_Quiz:	4
Log-In_Sign-Up:.....	4
Merch-Shop:	4
News:	4
Q&A:.....	5
style:.....	5
„index.html“:.....	5
Einheitliche Navigationsleiste	6
Entwurf:	6
Implementation:	6
<p id='credentials'>.....	6
Login- bzw. Anmelde-Webseite	7
Entwurf:	7
Implementation:	8
Q&A-Forum	9
Entwurf:	9
Implementation:	9
Merch-Shop.....	12
Entwurf:	12
Implementation:	12
News-Webseite	15
Entwurf:	15
Implementation:	15
Quiz	17
Entwurf:	17
Implementation:	18
<canvas> & Gauss Calculator	19
Entwurf:	19
Implementation:	20

,gauss.js':	21
,canvas.js':	21
,index.html'	23
Implementation:	23
Testen.....	24
Platzknapper Fehler (Gauss Calculator)	24
Unerklärter Fehler (Gauss Calculator)	24
Weitere Tests des Projekts	25
Hinweise und Bemerkungen	26

Technische Anforderungen:

- Verlangt ist eine **einheitliche Navigationsleiste**, die in allen Webseiten eingehalten wird,
- eine **Login- bzw. Anmelde-Webseite**,
- ein **Q&A-Forum**,
- ein **Merch-Shop**,
- eine Webseite, die **Neuigkeiten** anzeigt,
- eine **Quiz** Webseite,
- eine Webseite, die die dynamische Verwendung des **<canvas>-Elementes** vorweist,
- die zu benutzenden Sprachen sind **HTML5, CSS3 und JavaScript**,
- der Code ist auf ein **Niveau** eines **Programmiereinsteigers** zu halten, aber es soll trotzdem eine Anzahl von Einführungsbeispielen für das Lernen und Lehren erbringen,
- **komplexe Codeausschnitte** können **ausgelassen** werden. Damit ist gemeint, dass das umfangreiche Validieren eines angelegten Benutzers oder ein aufwendiger Prozess, das Daten bearbeitet, ausgelassen werden kann.
- die Webseiten sind **browserbasiert**, da keine Serverseitige Sprache (wie PHP) angegeben ist.

Aufteilung des Projektes:

Die Struktur des gesamten Projektes kann sich allgemein in 6, bzw. 7 Ordnern unterteilen, wenn man die global-benutzten CSS3-Dateien beachtet. Im Folgenden werden die Inhalte der Ordner grob erläutert.

Bemerkung: Detaillierte Erklärungen von Hauptprozessen werden in den jeweiligen Entwurf- und Implementationabschnitten deutlicher erklärt

Gauss_Calculator:

- „gauss_calculator.html“: Als erstes ist die **Navigationsleiste** zu vermerken. Danach die **Tabelle**, in der man die Matrix von drei linearen Gleichungen eingeben kann. Als nächstes findet man ein Element, wo die Lösungen bzw. die Ergebnisse der Matrix ausgegeben werden. Als letztes findet man ein **leeres <canvas>-Element**.
- „canvas.js“: kontrolliert das Verhalten des **<canvas>-Elementes** in „gauss_calculator.html“.
- „gauss_round.png“: ist ein rundes Bild, das im **<canvas>-Element** eingesetzt wird.
- „gauss.js“: kontrolliert die Manipulation der **Tabelle** für das Eintragen der linearen Matrix Gleichungen.

Gauss_Quiz:

- „gauss_quiz.html“: Das erste Element ist die **Navigationsleiste**. Im Element ‚container‘ sind die **Fragen und deren Antworten** zu vermerken. Darauf folgt ein Element, in dem die **Ergebnisse** einer durchlaufenen Fragerunde eingeblendet werden sollen. Der Wert des Elementes <progress> dient nur als Beispiel.
- „gauss_quiz.css“: setzt den **Design** für die gesamte Datei ‚gauss_quiz.html‘.
- „answers.css“: setzt den **Design** für die jeweiligen **Antwortmöglichkeiten** in ‚gauss_quiz.html‘ fest. Diese Antwortmöglichkeiten sind in dem Element ‚questions_answers‘ enthalten.
- „buttonClick.js“: kontrolliert das designtechnische Verhalten der <button>-Elemente in dem Element ‚questions_answers‘.

Log-In_Sign-Up:

- „log-in_sign-up.html“: Die **Navigationsleiste** ist wieder an der ersten Stelle, danach folgt ein Element mit den **Einlogge- und Anmelde-Subelementen**. Als letztes ist eine kleine **Ansammlung von Nachrichten** zu finde, die den Benutzer informiert, ob er erfolgreich eingeloggt oder angemeldet ist.
- „log-in_sign-up.js“: kontrolliert alle HTML-Manipulationen in ‚log-in_sign-up.html‘.
- „log-in_sign-up.css“: setzt den **Design** der Datei ‚log-in_sign-up.html‘ fest.

Merch-Shop:

- „merch-shop.html“: Die **Navigationsleiste** ist wieder an erster Stelle, danach folgt eine **dateibezogene Navigationsleiste**, die den Benutzer zu seinem Warenkorb führt. Dann erkennt man ein Element mit Artikel, bestehend aus zwei **Artikel Gruppen (Hoodies und Masken)**. Danach folgt der **Warenkorb**.
- „merch-shop.css“: legt den **Design** für die gesamte Datei ‚merch-shop.html‘ fest.
- „merch-shop.js“: kontrolliert hauptsächlich welche **Artikel** in den **Warenkorb eingefügt** und welche **entfernt** werden.
- „purchase.html“: eine kleine Landewebseite, die den Benutzer über die Fortsetzung seiner Bestellung informiert.
- Zu finden sind noch die **Ordner Hoodies** und **Masks**. Diese beinhalten die Bilder jeglicher Artikel.

News:

- „news.html“: An erster Stelle ist wieder die **Navigationsleiste** zu finden. Dann folgt eine **Tabelle**, die hauptsächlich **zwei <input type="datetime-local">-Elemente** beinhaltet. Diese dienen zur Sortierung von Nachrichtenreportagen. Danach folgt ein Element mit den **jeweiligen Artikeln**.
- „news.css“: setzt den Design der gesamten Datei ‚news.html‘ fest.
- „news.js“: dient zur HTML-Manipulation in der Datei ‚news.html‘.

Q&A:

- „q&a.html“: Auch hier steht die **Navigationsleiste** an erster Stelle. Danach folgt ein Element, wo der Benutzer **Fragen in das Forum** stellen werden kann (diese Funktionalität wurde aus der Webseite entworfen, da sie überflüssig ist). Und natürlich folgt auch ein Element, das alle Reportagen beinhaltet („questions“).
- „q&a.css“: setzt den Design für die Datei „q&a.html“ fest, sowohl auch welche Design-features verändert werden, wenn der Bildschirm eine bestimmte Breite unterschreitet.
- „q&a.js“: dient als HTML-Manipulation für die meisten <button>-Elemente.

style:

- „credentials.css“: setzt den Design für das <p>-Element „credentials“. Dieses Element ist am Ende jeder HTML-Datei zu finden. Dies soll die Identifikation des Projekterstellers kennzeichnen.
- „navigation-bar.css“: setzt den Design der gewünschten Navigationsleiste am Kopf jeder Webseite (bitte Kopf nicht mit <head>-Element verwechseln. Hier ist der Kopf der im Browser angezeigte Webseite gemeint).

„index.html“:

- Diese Datei ist erstellt worden, um dem Entwickler die Startwebseite anzugeben.

Im Folgenden wird der Entwurf und die Implementation der technischen Anforderungen beschrieben.

Einheitliche Navigationsleiste

Entwurf:

- Als Entwurf soll ein einfacher <nav>-Element verwendet werden. Der href/link jedes anchor-Tags soll zu jeder anderen Webseite führen, die im Zusammenhang mit dem Projekt steht.

```
<nav>
  <a href='../Log-In_Sign-Up/log-in_sign-up.html'>Log-In/Sign-Up</a>
  <a href='../Gauss_Calculator/gauss_calculator.html'>Gauß Rechner</a>
  <a href='../Gauss_Quiz/gauss_quiz.html'>Gauß Quiz</a>
  <a href='../News/news.html'>News</a>
  <a href='../Merch_Shop/merch-shop.html'>Merch-Shop</a>
  <a href='../Q&A/q&a.html'>Q&A</a>
</nav>
```

Implementation:

- Jede Navigationsleiste wird von der Datei ,navigation-bar.css' die Designvorlage aufgetragen bekommen. Unten nochmal beschrieben.
- Die Navigationsleiste ist in jeder HTML-Datei am Anfang des body-Tags zu finden.
- Ebenfalls wird das Attribut class(Wert: ,active') dem <a>-Element zugeordnet, der die jeweilige Ansicht der Webseite anzeigen soll.
- Auch ein Logo ist hinzugefügt worden, um den Design aller Webseiten zu vereinheitlichen.
- Zu beachten ist auch, dass die <a>-Elemente, die den URL ändern können, nun in ein <div>-Element eingebunden sind, dass das Attribut class(Wert: ,header-right') bekommt. Das dient zur vereinfachten CSS-Gestaltung. Somit werden die funktionalitätsfähigen <a>-Elemente zu der rechten Seite der Webseite verschoben.
- Das Logo ist in jeder Webseite gleich. Es wird durch das <a>-Element mit dem Attribut class(Wert: ,logo') gekennzeichnet.
- Im Folgenden wird ein Beispiel gezeigt. Die Navigationsleiste ist aus der Datei ,gauss_calculator.html', wie man an dem Attribut class(Wert: 'active') erkennt.

```
<nav class="header" id="main_header">
  <a class="logo">Gauß.com</a>
  <div class="header-right">
    <a href='../Log-In_Sign-Up/log-in_sign-up.html'>Log-In/Sign-Up</a>
    <a href='../Gauss_Calculator/gauss_calculator.html' class="active">Gauß Rechner</a>
    <a href='../Gauss_Quiz/gauss_quiz.html'>Gauß Quiz</a>
    <a href='../News/news.html'>News</a>
    <a href='../Merch_Shop/merch-shop.html'>Merch-Shop</a>
    <a href='../Q&A/q&a.html'>Q&A</a>
  </div>
</nav>
```

<p id='credentials'>

- Dieses Element stellt nur einen Text dar, der den Entwickler verzeichnet. Die zugehörige CSS-Datei ist ,credentials.css' (unter ,style/credentials.css' zu finden).
- Das Element wird in jeder Webseite vorkommen und gilt somit auch als einheitliches Element.

```
<p id="credentials">
  © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
</p>
```

Login- bzw. Anmelde-Webseite

,log_in-sign_up.html'

Entwurf:

- Wie schon erwähnt beinhaltet jede Datei die einheitliche Navigationsleiste. Auch hier ist es der Fall.
- Als Kern der Webseite wird es einen Login- und ein Anmeldeformular geben.
- Man kann zwischen den Formularen mit Hilfe von <button>-Elementen wechseln.
- Zuletzt wird es eine Box geben, wo die Nachricht angezeigt wird, ob der Benutzer sich richtig eingeloggt hat oder erfolgreich ein Benutzerkonto erstellt hat. Diese Nachrichten werden in dem <div>-Element ,messages' eingefügt.
- Es ist zu beachten, dass die <button>-Elemente und die restlichen Elemente für das Einloggen und das Anmelden über JavaScript-Funktionen eingeblendet und ausgeblendet werden. Dafür wird das Attribut style (jeweils mit dem Wert: 'display:none') benutzt. Dies wird folgendermaßen geschehen: wenn man sich anmelden möchte wird das Anmelde-Formular angezeigt und das Login-Formular wird ausgeblendet. Andererseits wird das Login-Formular angezeigt und das Anmelde-Formular wird ausgeblendet. Diese Funktionalität ist JavaScript-gesteuert und wird näher in der Dateibeschreibung ,log_in_sign-up.js' erläutert.
- Das Folgende Bild zeigt einen groben Entwurf der geplanten HTML-Datei. Weitere Elemente werden im Laufe der Implementation entwickelt.

```
<body>
  <nav class="header" id="main_header">
    <a class="logo">Gauß.com</a>
    <div class="header-right">
      <a href='../Log-In Sign-Up/log_in_sign-up.html' class="active">Log-In/Sign-Up</a>
      <a href='../Gauss Calculator/gauss_calculator.html'>Gauß Rechner</a>
      <a href='../Gauss Quiz/gauss_quiz.html'>Gauß Quiz</a>
      <a href='../News/news.html'>News</a>
      <a href='../Merch Shop/merch-shop.html'>Merch-Shop</a>
      <a href='../Q&A/q&a.html'>Q&A</a>
    </div>
  </nav>

  <div id="mainContent">
    <div id="buttons">
      <div id="log-in-buttons">
        <button></button>
      </div>

      <div id="sign-up-buttons" style="display:none">
        <button></button>
      </div>
    </div>

    <div id="containers">
      <div id="log-in-container"></div>

      <div id="sign-up-container" style="display:none">
        </div>
    </div>

    <div id="messages">
      <p id="successfulLogIn">successful Log In</p>
      <p id="successfulSignUp">successful Sign Up</p>
    </div>

    <p id="credentials">
      © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
    </p>
  </div>
</body>
```


Implementation:

- Zusammenhang zwischen ,buttons' und ,containers'
 - In dem ,log-in-buttons'-Element sind zwei <button>-Elemente zu finden. Das <button>-Element ,submit-log-in' ist für das Anzeigen des Elementes ,succesfulLogIn' und das Ausblenden des Elementes ,succesfulSignup' da. Das <button>-Element ,sign-up' ist für das Ausblenden des ,log-in-containers' sowie des Elementes ,log-in-buttons' da. Zur selben Zeit blendet es die Elemente ,sign-up-container' und ,sign-up-buttons' ein.
 - Die selbe Logik, doch mit dem umgekehrten Sinn, ist in dem ,sign-up-buttons'- und ,sign-up-container'-Elementen zu finden. Das Element ,submit-sign-up' ist für Anzeigen des Elementes ,succesfulSignUp' und das Ausblenden des Elementes ,succesfulLogIn' da. Das <button>-Element ,log-in' ist für das Ausblenden des ,sign-up-containers' sowie des Elementes ,sign-up-buttons'. Zur selben Zeit blendet es die Elemente ,log-in-container' und ,log-in-buttons' ein.
- In den Elementen ,sign-up-container' und ,log-in-container' sind <form>-Elemente enthalten. Diese beinhalten Optional- und Pflichteingabefelder für die korrekte Anmeldung oder Erstellung eines neuen Benutzers.
- Wieso befinden sich die Eingabefelder unter den <button>-Elementen?
 - Das hat einen designtechnischen Grund. Bei einer Browserfensterbreite unter 840px werden die <button>-Elemente nach oben versetzt, was dem Benutzer eine intuitive Reihenfolge der zu benutzenden Elemente (wie das Ausfüllen von Feldern, etc.) anbietet.
 - Weitere designtechnische Manipulationen befinden sich in der Datei ,log-in_sign-up.css'

Q&A-Forum

,q&a.html'

Entwurf:

- Als Leitbeispiel habe ich den YouTube Kommentarbereich gewählt. Er ist sehr einfach gestaltet und das Benutzen ist ebenfalls intuitiv. Statt den Begriff ‚Kommentar‘ werde ich natürlich ‚Frage‘ in der Dokumentation verwenden.
- Es gibt ein Element, wo der aktuelle Benutzer seine Frage schreiben, verwerfen und schicken kann. Das wird an der obersten Stelle im Kern der Webseite sein.
- Des Weiteren findet man das <section>-Element ‚questions‘. Es dient als Ansammlung von gestellten Fragen.
- Das Sortieren der Kommentare ist bei YouTube komplexer gestaltet als in der Anforderung verlangt, deshalb werde ich die Fragen nach dem Sendedatum rückläufig sortieren. Das heißt, dass die neuesten Kommentare in der Ansammlung am Anfang eingebunden sind.
- Selbstverständlich binde ich auch die Navigationsleiste an erster Stelle und den ‚credentials‘-Paragraf an letzter Stelle ein.

```
<body>
  <nav class="header" id="main_header">
    <a class="logo">Gauß.com</a>
    <div class="header-right">
      <a href='../Log-In_Sign-Up/log-in_sign-up.html' id="Log-In/Sign-Up">Log-In/Sign-Up</a>
      <a href='../Gauss Calculator/gauss_calculator.html'>Gauß Rechner</a>
      <a href='../Gauss Quiz/gauss_quiz.html'>Gauß Quiz</a>
      <a href='../News/news.html'>News</a>
      <a href='../Merch Shop/merch-shop.html'>Merch-Shop</a>
      <a href='../Q&A/q&a.html' class="active">Q&A</a>
    </div>
  </nav>

  <section id="askQuestion">
  </section>
  <section id="questions">
    <div id="question 2">
    </div>
    <div id="question 1">
    </div>
  </section>
  <p id="credentials">
    © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
  </p>
</body>
```

Implementation:

- Im Folgenden wird die Struktur eines Frage-Objektes erklärt. Grundgenommen besteht eine gestellte Frage aus einem Element, das die Informationen der Frage selbst beinhaltet, mit der id ‚container‘ gekennzeichnet, und einem Element, das die Antworten der Benutzer beinhaltet, mit der id ‚answers_box‘ gekennzeichnet.
- ‚container‘:
 - ‚mainData‘: hier befinden sich die signifikanten Daten des Frage-Objektes (Uhrzeit und Datum der Erstellung, Benutzername und die Frage selbst).
 - ‚opinions‘: hier sind die Anzahlen von Likes und Dislikes zu finden.
 - ‚give_answer‘: dies ist ein Element, in dem man eine Antwort geben kann.

- ,answers_box':
 - Dieses Element dient als Pool von Antworten, das nach dem Sendedatum rückläufig sortiert wird.
 - Die Subelemente sind natürlich die einzelnen Antworten. Auch diese haben eine gewisse Struktur.
 - Wie auch bei einer Frage hat eine Antwort die signifikanten Daten im Subelement ,mainData' (diese Daten sind analog Frage-Objekt Daten vergleichbar) und
 - eine Antwort beinhaltet Meinungsäußerungen, die bei den Likes und Dislikes, angezeigt werden.
- Das Frage-Objekt wird in der Webseite mit einem roten dicken Rand markiert.

```
<section id="askQuestion">
  <p>ask a Question</p>
  <textarea type="text" id="question"></textarea>
  <br><input type="checkbox" id="anonymous">
  <p style="display:inline">Anonymous</p>
  <br><button>ask question</button>
</section>
<section id="questions">
  <div id="question 2" class="question">
    <div id="container">
      <div id="mainData" class="mainData">=
      <div id="opinions">=
      <div class="give_answer">=
    </div>
    <!-- answers corresponding to the question -->
    <p onclick="changeState(this.parentElement)" class="answersCounter">View <span id="answers_amount">
      <!-- javascript generated -->
    </span>Answers</p>
    <div id="answers_box" class="close">
      <div id="answer_1" class="answer">
        <div id="mainData">=
        <div id="opinions">=
      </div>
      <div id="answer_2" class="answer">=
    </div>
  </div>
  <div id="question 1" class="question">=
</section>
```

- Wie funktioniert das liken und disliken:
 - Diese Funktionalität wird von der JavaScript Funktion ,opinion()' gesteuert (unter Q&A/q&a.js zu finden).
 - Jedes <button>-Element, das eine Like- oder Dislike-Funktionalität hat, beinhaltet ein -Element, jeweils mit der id ,amount' gekennzeichnet, und ein <input>-Element, jeweils mit der id ,used' gekennzeichnet.
 - Um dies zu verstehen muss man das <button>-Element, das eine Like- oder Dislike-Funktionalität hat, verstehen.
 - Das Element ,amount': ist eine Werteangabe, die mitverfolgt wie viele Benutzer das jeweilige <button>-Element angeklickt haben.
 - Das Element ,used': gibt an, ob das jeweilige <button>-Element schon angeklickt ist oder nicht. Achtung, der Wert hat eine boolesche Repräsentation, ist aber ein String.
 - Das folgende Bild zeigt ein Beispiel, bei dem es sich um ein Like-<button>-Element handelt. Dies erkennt man an dem id-Attribut.

```
<button id="likes" onclick="opinion(this)">
  likes <span id="amount">3</span>
  <input type="hidden" id="used" value=false>
</button>
```

- Wie man sieht wird die Funktion ,opinion()' aktiviert, wenn man das <button>-Element anklickt.
- opinion():
 1. Wenn das Element ,used' den Wert ,false' aufweist, so erhöht man den Wert von ,amount', da es sich um ein noch nicht angeklicktes <button>-Element handelt.
 2. Wenn das Element ,used' den Wert ,true' aufweist, so senkt man den Wert von ,amount'. Hier würde es sich um ein schon angeklicktes <button>-Element handeln, das nochmal angeklickt worden ist. Dies weist darauf hin, dass der Benutzer den Status widerrufen möchte.
 3. Wenn das ,used' Subelement des gegenseitigen <button>-Elementes den Wert ,true' aufweist, so geschieht folgendes:
 - i. der Wert von ,amount' wird um eins verringert und
 - ii. der Wert von ,used' wird auf ,false' eingestellt, was angibt, dass das gegenseitige Element nicht aktiv ist.
- <button>-Elemente mit Like- bzw. Dislike-Funktionen, die auch von dem Benutzer angeklickt worden sind, werden mit dem Attribut class(Wert: ,active') gekennzeichnet. Das Attribut class(Wert: ,active') gibt dem Element einen blauen Hintergrund. Die Definition dessen Attributes findet man in der Datei ,style/navigation-bar.css'

Merch-Shop

,merch-shop.html'

Entwurf:

- Die oberflächliche Struktur des Merch-Shops besteht aus einem Element, das die Waren/Artikel anbietet (mit ,container' gekennzeichnet) und einem Element, das die ausgewählten Artikel beinhaltet (mit ,cartContainer' gekennzeichnet).
- Im folgenden Bild kann man sehen, dass der ,container' schon zwei Artikelgruppen hat. Das ist einmal die Artikelgruppe Hoodies und einmal Masken (mit ,Hoodies' und ,Masks' gekennzeichnet).

```
<body>
  <nav class="header" id="main_header">
    <a class="logo">Gauß.com</a>
    <div class="header-right">
      <a href='../Log-In_Sign-Up/log-in_sign-up.html' id="Log-In/Sign-Up">Log-In/Sign-Up</a>
      <a href='../Gauss Calculator/gauss_calculator.html'>Gauß Rechner</a>
      <a href='../Gauss Quiz/gauss_quiz.html'>Gauß Quiz</a>
      <a href='../News/news.html'>News</a>
      <a href='../Merch Shop/merch-shop.html' class="active">Merch-Shop</a>
      <a href='../Q&A/q&a.html'>Q&A</a>
    </div>
  </nav>

  <div id="container">
    <div id="Hoodies">
    </div>
    <div id="Masks">
    </div>
  </div>

  <div id="cartContainer">
  </div>
  <p id="credentials">
    © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
  </p>
</body>
```

Implementation:

- Im weiteren Verlauf der Dokumentation werde ich den Begriff „Warenkorb“ benutzen, um auf das Element ,cart' hinzuweisen.
- Es ist zu erwähnen, dass eine weitere Navigationsleiste eingeführt wurde, die zum ,cartContainer' hinweist. Diese Leiste ist nur für den Merch-Shop erstellt worden.

```
<!-- this navigation bar is specifically made for the n
<nav class="header" id="sub_header">
  <div class="header-right" id="sub_header_anchors">
    <a href="#cartContainer">
      <!-- onclick="if (document.getElementById('
      Cart
    </a>
  </div>
</nav>
```

- Die umfangreichste Struktur bzw. das umfangreichste Objekt in ‚merch-shop.html‘ ist das Artikel-Objekt. Des Weiteren werde ich das Objekt erläutern und erklären wie es funktioniert.
- Jedes Artikel-Objekt besteht aus:
 - o einem -Element, das ein Bild des Artikels anzeigt,
 - o einem <p>-Element, das den Artikelpreis anzeigt,
 - o einem <button>-Element (jeweils mit der id ‚add‘ markiert), das den Artikel als ausgewählt markiert und in den Warenkorb platziert, und
 - o einem <button>-Element (jeweils mit der id ‚remove‘ markiert), das den Artikel aus dem Warenkorb entfernt und nicht mehr als ausgewählt markiert.
 - o Zu vermerken ist noch, dass jeder Wert des Attributes ‚id‘ intuitiv hergeleitet ist und nicht willkürlich erstellt worden ist. Die Herleitung basiert zur Hälfte von dem Kürzel des Artikels selbst und zur anderen Hälfte aus der Reihenfolge.
 - o In dem folgenden Bild sieht man das Beispiel eines Artikel-Objektes. Es handelt sich um den ersten Hoodie.

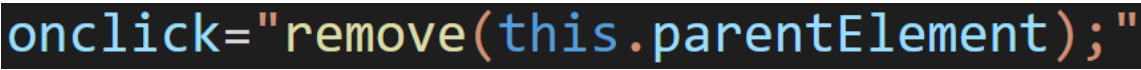
```
<div id='H1' class="notInCart">
  
  <br>
  <p id='price' value='14'>Price:14 Euros</p>
  <button id="add" onclick="add(this.parentElement);">Add to Cart</button>
  <button id="remove" onclick="remove(this.parentElement);">Remove from Cart</button>
</div>
```

- o Der Warenkorb ist mit der id ‚cart‘ vermerkt. Die nebensächlichen Elemente um ‚cart‘ dienen zur Steigerung der Benutzerfreundlichkeit der Webseite. Zum Beispiel ist ein <a>-Element eingefügt worden, um das Fenster des Benutzers schneller zu der Navigationsleiste zu führen.

```
<a href="#main_header">to the top <!-- brings you to the top of the page, where the main navigation bar starts --></a>
<div id="cartContainer" class="emptyCart">
  <a href="purchase.html"> purchase Items!</a>
  <div id="cart">
    <!-- the actual cart -->
  </div>
  <a href="purchase.html"> purchase Items!</a>
  <p id="emptyMessage">Your cart is currently empty</p>
</div>
<a href="#main_header">to the top</a>
```

- Wie funktioniert das Platzieren eines Artikels in und Deplatzieren eines Artikels aus dem Warenkorb?
 - o Dies kontrollieren die JavaScript Funktionen ‚add()‘ und ‚remove()‘. Beide können in der Datei ‚Merch Shop/merch-shop.js‘ gefunden werden.
 - o Das Platzieren kontrolliert die JavaScript Funktion ‚add()‘. Im Folgenden Abschnitt wird sie beschrieben.
 - Wenn der Benutzer das <button>-Element ‚add‘ anklickt, wird diese Funktion ausgelöst.
 - Als Argument, wird der referenzierte Artikel-Objekt angegeben. Dieser Artikel-Objekt wird von dem <button>-Element spezifiziert. Im folgenden Bild ist ein Beispiel.

```
▪ onclick="add(this.parentElement);"
```

- Als erster wichtigster Schritt wird der Artikel geklont. Das Klonen des Artikels ist wichtig, um nicht den referenzierten Artikel in den Warenkorb zu verschieben, sondern eine Repräsentation.
- Dieser Klon wird in das Element ‚cart‘ eingefügt.
- Hinweis: wenn man die ‚[HMLObject].cloneNode(true)‘ Funktion aufruft, so wird das gesamte Objekt geklont. Das bezieht auch die Unterobjekte (children).
- Das Deplatzen kontrolliert die JavaScript Funktion ‚remove()‘. Im Folgenden Abschnitt beschrieben:
 - Wenn der Benutzer das <button>-Element ‚remove‘ drückt, wird diese Funktion ausgelöst.
 - Als Argument, wird der referenzierte Artikel-Objekt angegeben. Dieser Artikel-Objekt wird von dem <button>-Element spezifiziert. Im folgenden Bild ist ein Beispiel.
- 

```
onclick="remove(this.parentElement);"
```
- Als erster wichtigster Schritt wird das Artikel-Objekt von dem ‚container‘ und die jeweilige Repräsentation aus dem Warenkorb ausgewählt. Dies ist wichtig, um das richtige Artikel-Objekt zu entwerfen, wie man gleich erkennen wird.
- Das Artikel-Objekt im ‚container‘ bekommt das Attribut class(Wert: ‚notInCart‘). Dies entfernt den Streifen des Artikel-Objektes, was im Endeffekt das Objekt als nicht ausgewählt kennzeichnet.
- Die Repräsentation aus dem Warenkorb wird nun aus dem HTML-Dokument gelöscht.

News-Webseite

,news.html'

Entwurf:

- Die oberflächliche Struktur der News-Webseite besteht aus einem Element, das die Neuigkeiten/Reportagen filtert (mit ,filterByDate' gekennzeichnet) und einem Element, das die ausgewählten Neuigkeiten beinhaltet (mit ,allNews' gekennzeichnet).
- Die HTML-Objekte, die die Informationen der Neuigkeiten beinhalten werden mit einem intuitiven ,id' Attribut gekennzeichnet. Auch hier besteht das Attribut aus dem Objektnamen (in diesem Fall wurde das englische Wort ,article' benutzt) und der Reihenfolge (abwärts gezählt)
- Im folgenden Bild kann man sehen, dass ,allNews' schon 4 Neuigkeiten hat. Ebenfalls erkennt man das die Neuigkeiten-Objekte abwärts gezählt werden. Damit soll erreicht werden, dass die Neuigkeiten-Objekte, wie auch die Frage-Objekte im Q&A-Forum, nach dem veröffentlichten Datum sortiert werden.

```
<body>
  <nav class="header">

  <div id="filterByDate"></div>

  <section id="allNews">
    <!-- all the articles go in this section element -->
    <div id="article_4">
    </div>

    <div id="article_3">
    </div>

    <div id="article_2">
    </div>

    <div id="article_1">
    </div>
    <!-- have to put the article in a <div id=article>, because otherwise
  </section>

  <p id="credentials">
    © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
  </p>
</body>
```

Implementation:

- ,filterByDate'
 - In diesem Element befinden sich hauptsächlich zwei <input type='datetime-local'-Elemente. Damit ist es dem Benutzer möglich Neuigkeiten auszuwählen, die ab einer Zeit oder und bis einer Zeit veröffentlicht worden sind.
 - Zu beachten ist, dass die <input>-Elemente in einem <table>-Element stehen. Das dient nur zur Strukturierung des HTML-Codes und Webseitendesigns.

- ‚allNews‘
 - Wie schon erwähnt werden in diesem Element alle Neuigkeiten-Objekte aufgefunden.
 - In der folgenden Auflistung wird das Neuigkeiten-Objekt beschrieben:
 - es besteht aus einem <input>-Element (jeweils mit der ‚id‘ ‚publication‘ gekennzeichnet), das den Veröffentlichungszeitpunkt festhält,
 - einem <p>-Element, das den Titel der Reportage anzeigt und
 - einem <article>-Element, das den Inhalt der Reportage enthält.
- ```
<div id="article_3" class="notLiked">

<input type="text" id="publication" value="2021-01-08T16:15" readonly>
 <p onclick="showArticle(this.parentElement)">First Friday of the Year</p>
 <article id="article_content" class="hideContent">
 <button id="close" onclick="closeArticle(this.parentElement)">close</button>
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nam, optio nemo
 saepe iusto natus eum alias magnam facilis quam ducimus provident numquam
 adipisci possimus!
 </p>
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nam, optio nemo
 saepe iusto natus eum alias magnam facilis quam ducimus provident numquam
 adipisci possimus!
 </p>
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nam, optio nemo
 saepe iusto natus eum alias magnam facilis quam ducimus provident numquam
 adipisci possimus!
 </p>
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nam, optio nemo
 saepe iusto natus eum alias magnam facilis quam ducimus provident numquam
 adipisci possimus!
 </p>
 </article>
</div>
```
- Was ist mit ‚readonly‘ gemeint? Die ‚readonly‘ Einstellung verhindert eine Überschreibung des Wertes. Diese Einstellung ist nur bei den <input>-Elementen zu finden, da sie als Informationsträger und -vermittler für die Veröffentlichung der Neuigkeiten funktionieren. Die ursprüngliche Veröffentlichung soll erhalten bleiben und der Benutzer soll damit abgehalten werden dies zu verändern.
- Wie funktioniert das benutzerdefinierte Auswählen der Neuigkeiten?
  - Wie schon erwähnt wurde, hat der Benutzer die Möglichkeit seine Neuigkeiten, nach bestimmten Zeitpunkten auszuwählen. Einerseits kann man Neuigkeiten auswählen, die ab einem Zeitpunkt veröffentlicht worden sind, andererseits kann man Neuigkeiten auswählen, die bis zu einem Zeitpunkt veröffentlicht worden sind und man kann Neuigkeiten auswählen, die in einem Zeitfenster veröffentlicht worden sind.
  - Dazu kann man Zeitangaben in den Elementen ‚datetime-picker-from‘ und ‚datetime-picker-until‘ auswählen und das <button>-Element ‚filterAllNews‘ drücken. Dies aktiviert die Funktion ‚filterByDate()‘ (in ‚News/news.js‘ aufzufinden). Im Folgenden wird diese Funktion beschrieben.
  - ‚filterByDate()‘:
    - var validator: Der Inhalt dieser Variable ist eine Funktion, dessen boolescher Rückgabewert auf die angegebenen Zeiten des Benutzers basieren. Wenn der Benutzer zum Beispiel angibt ab welchem Zeitpunkt Neuigkeiten angezeigt werden sollen, so wird eine Funktion zurückgegeben, die überprüft ob das Veröffentlichungsdatum der Reportage über dem benutzerdefinierten Zeitpunkt steht (siehe ‚getValidator()‘). Der boolesche Wert verändert sich wenn man den Zeitpunkt angibt bis zu welchem die Reportagen sortiert werden sollen und natürlich wird auch überprüft ob ein Zeitintervall angegeben worden ist. Wenn

man keine Zeiten angegeben hat, so wird der boolesche Wert ‚true‘ für jedes Veröffentlichungsdatum zurückgegeben, was das Sortieren der Reportagen nicht beeinflusst.

- Als nächstes erkennt man eine for-Schleife. In ihr wird der Veröffentlichungsdatum nach den benutzerdefinierten Zeitpunkten geprüft. Eine wahre Prüfung ändert den Wert des Attributes ‚display‘ zu einem leeren String, was dazu führt, dass der Neuigkeiten-Objekt im Fenster angezeigt wird. Bei einer unwarhen Überprüfung wird das Attribut ‚display‘ zu ‚none‘ gesetzt, was dazu führt, dass das Neuigkeiten-Objekt im Fenster nicht angezeigt wird.

## Quiz

,gauss\_quiz.html‘

### Entwurf:

- Oberflächlich besteht die Quiz-Webseite an erster Stelle aus der einheitlichen Navigationsleiste, aus einem Element, das die Frage-Objekte beinhaltet (,questions\_answers‘), einem Element, das den Erfolg des Benutzers anzeigt (,results‘) und ebenfalls aus dem Element ,credentials‘.
- Die Frage-Objekte bestehen natürlich aus der formalen Frage, die in einem <h1>-Element beinhaltet sind und aus mehreren Antwortvorschlägen, von denen nur eins richtig ist. Diese Antwortvorschläge sind im Element ,single\_choice\_answers‘. Jede Antwortmöglichkeit befindet sich im <button>-Element.

```
<body>
 <nav class="header" id="main_header">=

 <div id="questions_answers">
 <section id="question1">
 <div id="singl_choice_answers">
 <button>...</button>
 <button>...</button>
 <button>...</button>
 <button>...</button>
 </div>
 </section>
 <section id="question2">=
 <section id="question3">=
 <section id="question4">=
 <section id="question5">=
 </div>

 <section id="results"></section>

 <p id="credentials">
 © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
 </p>
</body>
```

## Implementation:

- Im Folgenden wird das wiederholende Frage-Objekt erläutert. Im Bild sieht man das Frage-Objekt ,question3‘.

```
<section id="question3">
 <h1>Was war Gauß Haarfarbe?</h1>
 <div class="single-choice-answers">
 <button onclick="buttonClick(this)">Hauptsächlich grau</button>
 <button onclick="buttonClick(this)">blond</button>
 <button onclick="buttonClick(this)">türkis</button>
 </div>
 <ul class="navbar">

 PREVIOUS

 NEXT

 RESULTS

</section>
```

- Die formale Frage bleibt, wie auch im Entwurf erwähnt, im <h1>-Element.
- Als nächstes erkennt man eine Ansammlung an <button>-Elementen, die die Antwortmöglichkeiten anbieten. Beim Anklicken eines dieser Elemente wird der Hintergrund des Elementes blau. Diese Funktionalität wird in ,buttonClick()' kontrolliert (unter ,Gauss Quiz/buttonClick.js' zu finden).
- Wie funktioniert ,buttonClick()'?
  - o Die Funktion bekommt als Argument das <button>-Element, das von dem Benutzer angeklickt worden ist.
  - o Als nächstes sammelt es alle <button>-Elemente, die in dem Frage-Objekt vorhanden sind und setzt das Attribut ,class' zu einem leeren String. Das bewirkt, dass jegliche Definitionen, die sich auf das ,class'-Wert ,active' beziehen, ausgeschlossen werden. Im Endeffekt, wird der blaue Hintergrund des Elementes, falls es so sein soll, deaktiviert.
  - o Danach stellt es das Attribut ,class' des gedrückten <button>-Elementes auf ,active'. Dies bewirkt, dass der Hintergrund des Elementes blau wird. Die Definition dieses Attributes ist in ,style/navigation-bar.css' definiert.
- Als nächstes erkennt man das Element <ul>. Grundgenommen beinhaltet es zwei <a>-Elemente, dessen ,href'-Attribut zu Frage-Objekten zuweist, die in der Reihenfolge vorher- oder nachhergehen. Es kann auch vorkommen, dass eins dieser Elemente nicht vorhanden ist, da ein vorheriges oder nachheriges Frage-Objekt ebenfalls nicht vorhanden ist. Ein weiteres <a>-Element weist auf das ,results'-Element hin, wo die Ergebnisse einer Frage-Runde angezeigt werden.
- Im obigen Codeeinschnitt erkennt man, dass es sich um die dritte Frage in der Reihenfolge handelt. Es hat drei mögliche Antworten. Ebenfalls erkennt man, dass das vorhergehende Frage-Objekt das Element ,question2' ist und das nachgehende Frage-Objekt, das Element ,question4' ist.
- Das Überprüfen der Antworten und entsprechende Erstellen eines Resultates konnte eingefügt werden, wurde aber ausgelassen, da es zu einer komplexeren Codeimplementierung gekommen ist, was anforderungsmäßig zu vermeiden ist. Deshalb ist der Wert des Elementes <progress> nur auf einen fixen Wert eingestellt.

- Smoothscrolling: Wie man merkt sind im <head>-Element folgende <script>-Elemente enthalten.

```
<!-- smooth scrolling -->
<script src="https://cdn.jsdelivr.net/gh/cferdinandi/smooth-scroll@15.0.0/dist/smooth-scroll.polyfills.min.js">
</script>
<script>
 const scroll = new SmoothScroll('.navbar a[href*="#"]', {
 speed: 500
 });
</script>
```

- Ich werde das ausführliche Erklären der Elemente auslassen, da es einen viel zu großen Aufwand in Anspruch nehmen würde und ich mich selber mit der inneren Funktionalität nicht befasst habe.
- Der Zweck der <script>-Elemente besteht darin, den Übergang zwischen dem Klicken auf ein <a>-Element, das auf ein Element innerhalb des Fensters hinweist, und dem Verschieben des Fensters zum Zielelement, einige Zeit in Anspruch anzunehmen.

## <canvas> & Gauss Calculator

,gauss\_calculator.html'

### Entwurf:

- Auch hier wird an erster Stelle die Navigationsleiste und an letzter Stelle das ,credentials'-Paragraf eingefügt.
- Eine weitere Anforderung ist es das <canvas>-Element einzubinden. Dieses Element wurde in die Datei ,gauss\_calculator.html' eingebunden. Die Datei behandelt das Thema Gauß-Taschenrechner (engl.: Gauss Calculator). Das Gauss-Algorithmus kann eingefügt werden, aber auch hier habe ich darauf achten müssen, dass die Komplexität in den niedrig gehalten wird.
- Oberflächlich besteht die Webseite aus einem <table>-Element, das dem Benutzer die Möglichkeit gibt die Werte einer 3x4-Matrix einzugeben.
- Als nächstes sind zwei <button>-Elemente zu bemerken. Diese dienen zum Löschen der eingegebenen Werte und Berechnen der Werte. Natürlich wird keine echte Rechnung durchgeführt. Es wird nur ein Element eingeblendet, dass den Benutzer über das fiktive Berechnen der Matrix informiert.

```
<body contextmenu="menu">
 <nav class="header">
 <table id="gauss_table"></table>

 <button>clear</button>
 <button>solve</button>

 <div id="solutions"></div>

 <p id="credentials">
 © Copyright 2021 Victor Cislari (DHBW-Mannheim, Matrikel-Nr.: 8067138)
 </p>

 <canvas id="canvas"></canvas>
 </body>
```

- Ein <menu>-Element wird ebenfalls einbezogen. Zu beachten ist, dass das Element ab der HTML Version 5.2 nicht mehr unterstützt wird. Manche Browser, unter anderem Firefox, unterstützen

es jedoch weiter. Im Folgenden Bild sieht man, dass dieses Element in dem <head> definiert ist und im Element <body> als Attribut ,contextmenu' benutzt wird.

```
<head>
 <link rel="stylesheet" href="../style/credentials.css">
 <link rel="stylesheet" href="../style/navigation-bar.css">
 <title>Gauß Rechner</title>
 <menu type="context" id="menu">
 <menuitem label="clear">
 </menuitem>
 <menuitem label="solve">
 </menuitem>
 </menu>
</head>
```

### Implementation:

- Die Tabelle, in der die zu berechnende Matrix eingetragen werden soll, ist in dem Element ,divIdGaussTable' beinhaltet. Die Tabelle besteht natürlich aus dem Element <table>, in dem <input>-Elemente strukturiert platziert sind. Der Standardwert ist 0, wie man unter dem Attribut ,value' erkennt. Damit wird gesorgt, dass der Benutzer schnellere Eingaben tätigen kann und es wird verhindert, dass leere Werte übrigbleiben, was in späteren JavaScript Operationen zu unvorhersehbaren Unrichtigkeiten kommen könnte.

```
<div id="divIdGaussTable">
 <table id="gauss_table">
 <tr>
 <td><input type="number" value="0"></td>
 <td><input type="number" value="0"></td>
 <td><input type="number" value="0"></td>
 <td><input type="number" value="0"></td>
 </tr>
 <tr> ...
 </tr>
 <tr> ...
 </tr>
 </table>
</div>
```

- Als nächstes sind die zwei <button>-Elemente zu erkennen. Diese haben nun ,onclick'-Attribute. Beide bewirken die Funktion ,operate()', die in dem <head>-Element definiert ist.
  - o ,operate()': Diese Funktion bekommt aus dem Argument die Anweisung die Funktion ,calculate()' oder ,clear\_table()' aufzurufen. Mehr dazu unter der Implementation von der Datei ,gauss.js'. Am Ende wird die Funktion ,startAnimation()' aufgerufen. Dies bewirkt, dass für das <canvas>-Element einige Parameter neu geladen werden. Weitere technische Aspekte unter ,canvas.js'.
- Das Element ,solutions' beinhaltet ein <table>-Element, das eine Reihe von fiktiven Lösungsangaben beinhaltet. Wie schon erwähnt, ist eine Softwarelösung zur Berechnung von

Matrizen mithilfe des Gauß-Algorithmus möglich, jedoch muss man darauf achten nicht komplexen Code zu liefern, weshalb dieser Prozess entfällt.

- Es ist zu erwähnen, dass alle Elemente, außer `<canvas>`, in das Element `,mainContent'` eingefügt worden sind. Wieso? Wie später genannt wird, wird in dem `<canvas>`-Element das Hin- und Herspringen eines 2-dimensionalen Balles abgespielt. Dafür müssen Pixelangaben ausgelesen werden, die als Grenzen innerhalb des Fensters agieren. Dazu zählt die Fensterbreite als Seitengrenze, die Fenstertiefe, als Tiefengrenze und als Höchstgrenze wird die Tiefengrenze des Elementes vor dem `<canvas>`-Element. Alle charakteristischen Elemente, wie die Gaußmatrix oder der `,credentials'`-Paragraf, in dem Element `,mainContents'` einzufügen, soll Entwickeln das Weiterbilden der Webseite vereinfachen. Damit soll vor allem das Spezifizieren des Elementes vor dem `<canvas>`-Element in dem Skript `,canvas.js'` erspart werden.

```
<body contextmenu="menu">
 <div id="mainContent">...
</div>

 <canvas id="canvas"></canvas>
</body>
```

In den folgenden zwei Abschnitten werden die Skripte `,gauss.js'` und `,canvas.js'` erläutert.

### **`,gauss.js':`**

- `calculate()`: setzt das Attribut `,style'` des Elementes `,solutions'` zu einem leeren String, was bewirkt, dass das Element ohne jeglichen designtechnischen Angaben angezeigt wird.
- `clear_table()`:
  - o Es werden alle `<input>`-Elemente aus `,gauss_table'` in ein HTML-Objekt angesammelt, was wie ein Array von angefragten Elementen zu betrachten ist.
  - o Nun werden die Werte dieser Elemente auf „0“ gesetzt, was als Standardwert für das `<input>`-Element gilt.
  - o Als letztes wird das Element `,solutions'` ausgeblendet, indem man das Attribut `,style'` zu `,display:none'` setzt.

### **`,canvas.js':`**

- `startCanvasAnimation()`:
  - o In dieser Funktion werden alle benötigten Parameter für das `<canvas>`-Element gesetzt. Die Höhe, Breite, das eingesetzte Bild und die Startkoordinaten des angezeigten Bildes. Zum Schluss wird die Funktion `,startAnimation()'` aufgerufen, die das Bild durch das Fenster bewegt.
  - o Zu beachten ist, dass die Höhe des Elementes folgenderweise berechnet wird: von der Höhe des gesamten Fensters wird die Höhe des Elementes `,mainContent'` und die Breite der unteren Scrollleiste abgezogen. Ebenfalls ist die Breite des `<canvas>`-Elementes, die Breite des Fensters mit Absicht der Breite der Scrollleiste. Dies wird gemacht um den

Effekt zu erzeugen, dass das bewegende Bild in dem freien Feld des Fensters eingesperrt ist und gegen alle nächsten Elemente abspringt.

- Zum Schluss wird die Funktion `,startAnimation()'` aufgerufen. Dessen Funktionalität wird in dem Abschnitt `,startAnimation()'` erläutert.
- `getCoordinate()`:
  - Die Funktion speist einen Maximalwert ein und liefert eine mögliche Startkoordinate zurück. Die Startkoordinate gibt an, von welchem Breiten- oder Höhepunkt das Bild angezeigt wird.
  - Wieso `,Math.random()'`? Diese Funktion liefert einen willkürlichen Wert von 0 bis 1. Somit kann man eine willkürliche Koordinate innerhalb des Fensters bekommen. Dies geschieht, indem man die eingespeiste Koordinatenmaxime mit einem willkürlichen Wert zwischen 0 bis 1 multipliziert.
- `animate()`:
  - Das `,requestAnimationFrame()'` dient zur wiederkehrenden Animationsgestaltung. Es ist praktikabel `,requestAnimationFrame()'` in eine Variable zu speichern, mit der man im Falle eines Halteversuches die Animation beenden kann.
  - Unter dem Kommentar `,border-controll'` wird geprüft ob die Koordinaten des Bildes, die Rahmen des Fensters nicht überschreiten. Soll es zu solch einem Fall kommen, dann wird der entsprechende Richtungsvektor invertiert. Zuletzt wird die neue Koordinate um den angegebenen Richtungsvektor erweitert.
  - Um das Bild an neue Koordinaten zu rendern müssen folgende Prozesse getätigt werden. Das Bild wird gelöscht (Schritt 1) und dann wird das Bild mit den neuen Koordinaten in das `<canvas>`-Element eingefügt (Schritt 2).
- `startAnimation()`:
  - Diese Funktion wird aufgerufen, wenn das HTML-Dokument geladen hat oder wenn die `<canvas>`-Animation neu geladen werden muss.
  - Als erstes wird die derzeitige Animationsfunktion, die in der Variable `,cancel'` gespeichert wird, aufgehoben. Aus Versuchen wurde entdeckt, dass bei dem Auslass dieses Kommandos, der Richtungsvektor vergrößert wird, was dazu führt, dass die Schnelligkeit des bewegten Bildes positive beeinflusst wird. Heißt, die Animation wird schneller. Das ist jedoch nicht beabsichtigt.
  - Danach wird die Funktion `,reloadCanvasDimensions()'` ausgeführt. Sie liest neue Dimensionsangaben für die Breite und Länge des `<canvas>`-Elementes aus.
  - Zuletzt wird das Animationsgeschehen aktiviert, indem man die Funktion `,animate()'` aufrufend in die Variable `,cancel'` speichert. Somit kann man die Variable `,cancel'` für ein späteres Stoppen des Animationsgeschehen benutzen.
- `reloadCanvasDimensions()`:
  - Hier werden die Rahmen des Fensters ausgelesen.
  - Nützlich ist die Funktion, wenn die Breiten- oder Höhenangabe des Fensters sich verändert und eine sinngemäßes Rendering und Animationsvorgehen des Inhaltes von `<canvas>` ausgeübt werden soll.
  - Ohne diese Funktion könnte man das Inhalt des `<canvas>`-Elementes nicht in das erwünschte Fenster „einsperren“.
  - Die `,x'` und die `,y'` Variablen werden von der Funktion `,reloadCoordinates()'` gewertet.

- reloadCoordinates():
  - Als Argumente bekommt diese Funktion einen Wert, dass als Koordinateninstanz des rechten oder unteren Rand des Bildes zu interpretieren ist. Als zweites Argument wird entweder die derzeitige Fensterbreite oder –tiefe angegeben.
  - Falls der Rand des Inhaltes einen größeren Wert als der äußerste Rand des Fensters hat, so soll der äußerste Rand zurückgegeben werden. Andernfalls wird der aktuelle Rand zurückgegeben.
  - Damit soll sichergestellt werden, dass der Inhalt bzw- das Bild innerhalb des sichtbaren Fensters animiert wird.

,index.html‘

Implementation:

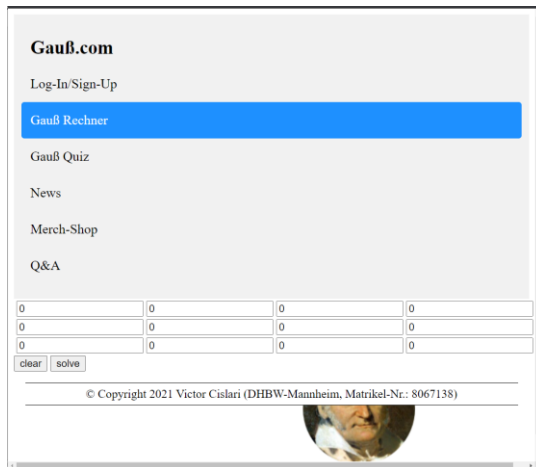
- Die außenstehende Datei soll bewirken, dass die Entwickler einen einfacheren Einstieg in das Testen der gesamten Webseite bekommen.
- Der Inhalt dieser Datei besteht nur aus der Navigationsleiste, damit man eine sofortige Übersicht von allen Webseiten hat, und des ,credentials‘-Paragraf. Diese Datei hat keinen Kern und thematisiert nichts.



## Testen

### Platzknapper Fehler (Gauss Calculator)

Bei dem Testen ist aufgefallen, dass das erzeugende Bild des `<canvas>`-Elementes, aus `,gauss_calculator.html'`, nicht jedes Mal richtig erscheint. Wenn dem Element weniger Breite oder Höhe angeboten wird, als 150px, so kommt es natürlich zu einem Problem, da das Bild mit einer Breite und Höhe von 150px je Seite, logischerweise nicht genug Platz hat. In einem solchen Fall, wird in der Funktion `,animate()'` (unter `,Gauss Calculator/gauss_calculator.html'` zu finden) der Richtungsvektor bei jeder neuen Animation wieder invertiert. Das bewirkt den Eintrag, dass das Bild über den Rahmen des `<canvas>`-Fensters erscheint und sich nur in die längliche Richtung des übergetretenen Rahmens bewegt. Im folgenden Bild ist zu erkennen, dass das Bild in `<canvas>` nicht genug Höhe zur Verfügung hat.

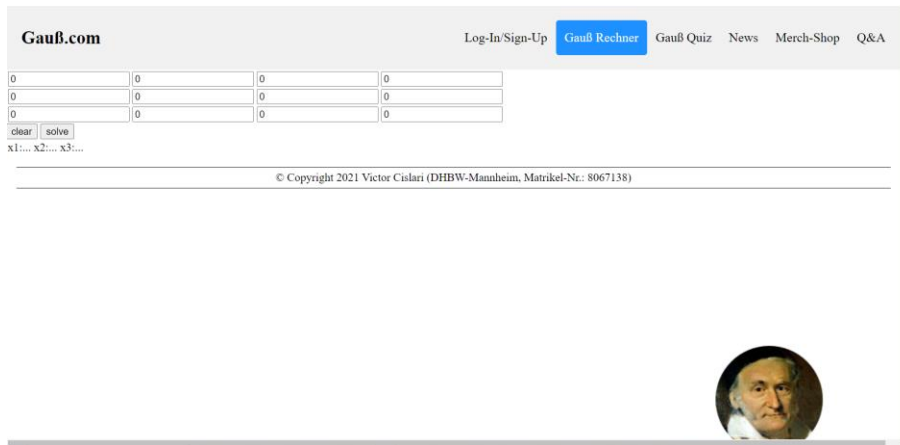


### Unerklärter Fehler (Gauss Calculator)

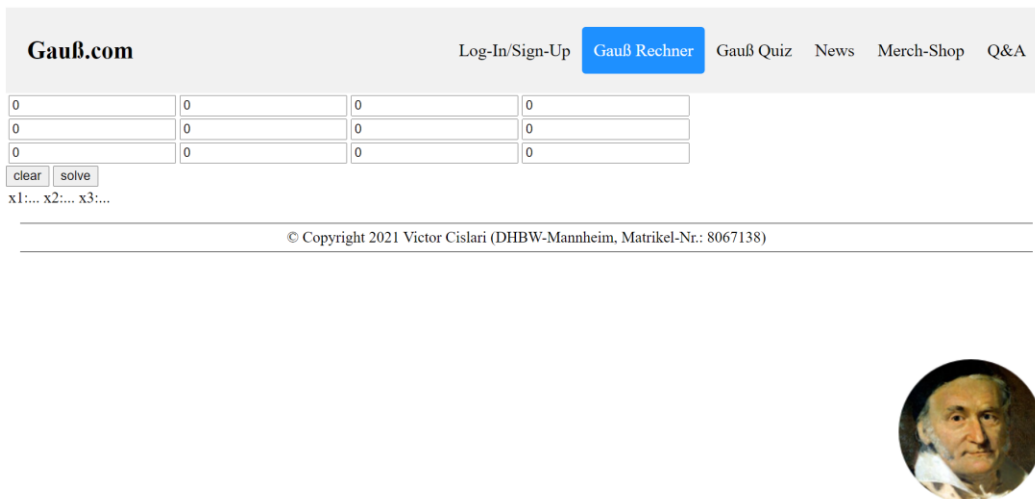
Ebenfalls taucht ein Fehler auf, wenn das Bild einen Rand anstößt und zur selben Zeit die Höhe des Elementes `,mainContent'` erhöht wird. Die Höhe wird erhöht indem der Benutzer sich das Element `,solutions'` anzeigen lässt. Dies kann bewirkt werden indem das `<button>`-Element `,buttonSolve'` angeklickt wird (für eine genaue Erklärung ist es empfohlen die Implementation der Datei `,gauss.js'` durchzulesen). In solch einem Fall unterschreitet das Bild den Rand und wird nicht über den unterschrittenen Rand neu animiert. Aus diesem `,Glitch'` habe ich es geschafft nur rauszukommen, wenn ich das `<button>`-Element `,buttonClear'` anklicke. Meine Vermutung ist, dass folgendes geschieht:

- das Element `,mainContent'` verringert sich,
- das Element `<canvas>` wird eine paar Pixel höher verschoben,
- die Parameter in der Datei `,canvas.js'` werden neu geladen, was dazu führt, dass eine höhere Fensterhöhe ausgelesen wird und
- dann wird in der Funktion `,animate()'` festgestellt, dass der untere Rand des Fensters weiter ist, als die Höhe des Bildes und alles läuft wieder wie es soll.

Im folgenden Bild ist ein Screenshot des Browsers zu sehen, wobei das Bild unterhalb der Scrollleiste animiert ist. Wenn man die Funktionen ‚startAnimation()‘ und ‚animate()‘ studiert, so erkennt man, dass dieses Problem nicht auftauchen sollte.



Lösungsansatz: ich habe versucht die Breite der Scrollleiste von den Variablen ‚canvasHeight‘ und ‚canvasWidth‘ abzuziehen (ich referenziere zu der Variablenmanipulation in ‚reloadCoordinates()‘). Leider kommt es immer noch zu dieser Störung. Im folgenden Bild sieht man, dass die Scrollleiste nicht im Fenster verfügbar ist, was man aber nicht sieht ist das „Zittern“ des Bildes in der unteren rechten Ecke. Mit dem „Zittern“ ist gemeint, dass die Funktion ‚animate()‘ die Richtungsvektoren immer wieder invertiert, wobei das Bild nicht aus den umgebenen Pixeln rausanimiert werden kann. Meiner Meinung nach kann es folgendes bedeuten: es wurde ausgelesen, dass die Breite und Höhe des <canvas>-Elementes größer sind, als die Koordinaten des rechten und unteren Rand des Fensters. Somit entsteht das Ständige Vektorinvertieren.



Leider bin nach mehreren Lösungsansätzen mit dem Problem nicht weitergekommen.

### Weitere Tests des Projekts

Auf Grund des nicht komplexen Codes, traten, nach der Implementation, keine weiteren Probleme bei den restlichen Webseiten auf.

## Hinweise und Bemerkungen

- Das Ein- und Ausblenden von Elementen kann auch mit dem Attribut ‚opacity‘ gesteuert werden. Ich habe mich für ‚display:none‘ entschieden, weil es meiner Meinung nach intuitiver zu verstehen ist.
- Mit „Kern der Webseite“ meine ich die charakteristischen Elemente der Webseite. Dazu gehört nicht das Paragraf ‚credentials‘ oder die Navigationsleiste.
- In jeder .html-Datei ist im <head>-Element die Metainformation angegeben, dass der ‚charset‘ auf „utf-8“ eingestellt ist. Dies bewirkt, dass deutsche Umlaute, wie „ä“, „ö“ oder „ü“, im Browser korrekt interpretiert werden. Man könnte auch HTML-Codes verwenden (&auml, &ouml, &uuml). Jedoch ist das sehr mühsam für jeden Umlaut einen HTML-Code zu benutzen.

```
<head>
 <meta charset="utf-8">
 <link rel="stylesheet" href="style/navigation-bar.css">
 <link rel="stylesheet" href="style/credentials.css">
 <title>Homepage</title>
</head>
```

- In allen HTML-Webseiten, außer ‚merch-shop.html‘, wurde von der Navigationsleiste die ‚mainHeader‘ id entfernt. Dieses Attribut hat sich im Nachhinein als unbrauchbar erklärt.
- ‚gauss\_calculator.html‘: leider habe ich keine Lösung gefunden die Fenstermaße auszulesen, während der Benutzer diese von den Browserrahmen aus verändert. Somit geschieht die Neuauslesung der Fensterbreite und -höhe für <canvas> nur, wenn der Benutzer das <button>-Element ‚buttonClear‘ oder ‚buttonSolve‘ tätigt.

```
<button onclick="operate('clear')" id="buttonClear">clear</button>
<button onclick="operate('solve')" id="buttonSolve">solve</button>
```

- Im Nachhinein ist eine letzte Veränderung zu vermerken. Die Navigationsleiste wurde um eine Verknüpfung erweitert. Mit der spähten Implementation der Datei ‚index.html‘ wurde beschlossen in jeder Navigationsleiste einen Link zu der Datei ‚index.html‘ einzufügen. Dies ist für alle Dateien zu vermerken. Diese spähte Implementation bringt keine Funktionalitätsänderung zu anderen Webseiten, somit gelten alle Beschreibungen in der Dokumentation. Der zu betätigende <a>-Element, um auf ‚index.html‘ zuzugreifen ist mit dem Attribut class(Wert: ‚active‘) vermerkt.

```
<nav class="header">
 Gauß.com
 <div class="header-right">
 Log-In/Sign-Up
 Gauß Rechner
 Gauß Quiz
 News
 Merch-Shop
 Q&A
 </div>
</nav>
```