Campus Tour Dokumentation

Die Dokumentation für die Webseite auf der die CampusTour der Uni Stutttgart läuft.
Hier sind alle wichtigen Funktionen aufgeführt um die CampusTour Webseite zu verwenden und zu verändern.

uniqUS, makeMINTcool

Inhaltsverzeichnis

CamusTour Documentation

Installation

Zurzeit wird die CampusTour über GitHub Pages gehostet. Erreichbar ist diese unter folgendem Link: CampusTour (https://aldammar.github.io/campusTour/)

Wenn man die CampusTour verändert will oder sie unter einem eigenen Link hosten will, dann gibt es verschiedene Möglichkeiten:

GitHub Pages

Hier kann man sowohl einen eigenen Link verwenden (oder den bestehenden) als auch die CampusTour verwenden.

GitHub Pages einrichten

1. **Repository forken:** Entweder direkt auf GitHub (https://github.com/Aldammar/campusTour/fork), über die GitHub Desktop App oder über die GitHub CLI. Weitere Informationen dazu gibt es hier (https://docs.github.com/de/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo).

2. **Repository klonen:** Das geforkte Repository kann nun geklont werden. Dies muss nicht gemacht werden, wenn die GitHub Desktop App oder die GitHub CLI mit `--clone=true` verwendet wurde. Zum Klonen kann man entweder die GitHub Desktop App (https://docs.github.com/de/repositories/creating-and-managing-repositories/cloning-a-repository?tool=desktop), die GitHub CLI (https://docs.github.com/de/repositories/creating-and-managing-repositories/cloning-a-repository?tool=cli) oder mit der Git Bash machen. Für die Git Bash wird der folgende Befehl benötigt:

   ```
   git clone https://github.com/YOUR-USERNAME/YOUR-REPOSITORY
   ```

3. **Änderungen vornehmen:** Nun können Änderungen an der CampusTour vorgenommen werden. Hierfür kann man die im How-To (How-Tos: Einfache Änderungen) beschriebenen Schritte verwenden. Wenn man mehr als die im How-

To (How-Tos: Einfache Änderungen) beschriebenen Sachen ändern will, dann kann man sich an die Referenz (Referenz) wenden.

4. **Änderungen hochladen:** Nachdem die Änderungen vorgenommen wurden, muss man diese committen und pushen, damit sie auch von GitHub erkannt werden.
5. **GitHub Pages aktivieren:** In den Repository-Einstellungen unter dem Tab **Pages** kann die GitHub Pages aktiviert werden. Hierbei muss man noch auswählen, von welchem Branch die GitHub Pages erstellt werden sollen. Standardmäßig ist dies der **main**-Branch im obersten Ordner. Mehr hierzu gibt es auf der GitHub Pages Seite (https://pages.github.com/).
6. **Fertig:** Nun sollte die CampusTour unter **https://YOUR-USERNAME.github.io/YOUR-REPOSITORY/** erreichbar sein.

Eigene Domain

Hier kann man alles so machen, wie bei den GitHub Pages, es ist aber etwas komplizierter, wenn man die Seite einfach nur hosten will.

> Man benötigt für diese Methode einen HTTPS-Server. Dieser kann entweder selbst aufgesetzt werden oder gemietet werden.

Selber hosten

1. **Server überprüfen:** Der Server muss HTTPS und JavaScript unterstützen. Die meisten Server kommen mit einer JavaScript-Unterstützung, jedoch nicht unbedingt mit einer HTTPS-Unterstützung. Sollte SSL nicht eingerichtet sein, finden sich im Internet viele Anleitungen dazu.
2. **Repository auf den Server laden:** Entweder lädt man den aktuellen Zweig direkt als zip herunter (GitHub (https://github.com/Aldammar/campusTour/archive/refs/heads/master.zip)) und entpackt dieses auf dem Server oder man erstellt wie unter GitHub Pages ("GitHub Pages" in "CamusTour Documentation") beschrieben ein Repository und cloned dieses auf den Server.
3. **Server konfigurieren:** Hierbei muss man darauf achten, dass der Pfad zum Ordner des Repositories auf dem **root**-Pfad des Servers liegt. Man kann dann über

> **https://YOUR-DOMAIN/PATH-TO-REPOSITORY-FROM-ROOT** auf die CampusTour zugreifen. Wenn man mit **https://YOUR-DOMAIN/** auf die CampusTour zugreifen will, dann muss man den Pfad zum Repository als **root**-Pfad des Servers setzen. Die genaue Konfiguration kommt auf den Server an.

Verwendung

Um die Webseite zu verwenden, benötigt man Zugriff auf die Kamera und beim ersten Laden Internet. Dann kann man die Webseite so, wie sie ist, verwenden.

How-Tos: Einfache Änderungen

Hier gibt es mehrere Beschreibungen für einfache Änderungen.

How To: Modelle ändern

Voraussetzungen

Stelle sicher, dass du dein Modell im *gLTF* Format hast! Folgende Dateien sind notwendig:

- **model.gltf**
- **model.bin**
- **model_normal.png**
- **model_roughnessMetallic.png**
- **model_baseColor.png**

> ⚠ Du kannst die ganzen Dateien auch gebündelt in einer **model.glb** Datei einbinden!

Schritte

Datei und Kennzeichnung ändern

Ändere Modell

Lade das Model von der Webseite herunter oder exportiere es aus dem Programm.

1. Kopiere oder verschiebe das Model mit allen Dateien in den Ordner **campusTour/models/**.
2. Suche in der Datei **ar.html** nach dem Tag `<a-assets>`

3. Ändere den Pfad des Modells und die Kennzeichnung (`<a-asset-item src="models/MODELL_NAME.[glb/gltf]" id="EINDEUTIGE_KENNZEICHNUNG_DES_MODELLS"`).
4. Suche nach der alten Kennzeichnung des Modells in der Datei und ersetze diese.

> ⚠ Du kannst für das Ändern der Kennzeichnung **refactoring** verwenden, wenn das dein Editor anbietet. Dann fällt der letzte Schritt weg

> ⚠ Wenn du nicht alle alten Kennzeichnungen änderst, dann könnte es sein, dass das Modell nicht an der richtigen Stelle angezeigt wird.

5. Verändere noch die Skalierung, Position und Drehung des Modells in den entsprechenden Markern, bei denen das Modell geändert wurde.

```
<a-marker>
    <a-entity
            gltf-model="#MODELL_NAME"
            scale="0 0 0.1"
            position="0 0 0"
            rotation="0 0 0"
    ></a-entity>
</a-marker>
```

Nur die Datei ändern

Ändere Modell Datei

Lade das Model von der Webseite herunter oder exportiere es aus dem Programm.

1. Kopiere oder verschiebe das Model in den Ordner **campusTour/models/**.
2. Suche in der Datei **ar.html** nach dem Tag `<a-assets>`

3. Ändere den Pfad des Modells (`<a-asset-item src="models/MODELL_NAME.[glb/gltf]"`)
4. Verändere noch die Skalierung, Position und Drehung des Modells in den entsprechenden Markern, bei denen das Modell geändert wurde.

```
<a-marker>
    <a-entity
            gltf-model="#MODELL_NAME"
            scale="0 0 0.1"
            position="0 0 0"
            rotation="0 0 0"
    ></a-entity>
</a-marker>
```

How To: Marker ändern

Voraussetzungen

Generiere dir auf der Seite AR.js Marker Training (https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html) einen Marker mithilfe eines Bildes. Auf folgendes solltest du achten:

- Der Marker sollte nicht rotationssymmetrisch sein
- Der Marker sollte nicht zu komplex sein

Schritte

Marker ändern

Lade von dem Marker Generator sowohl den Marker als auch das Bild herunter.

1. Kopiere oder verschiebe den Marker in den Ordner **campusTour/assets/patterns** und das Bild in den Ordner **campusTour/assets/images/**.

2. Suche den Marker, den du ändern willst und setze dort den Pfad auf deinen neuen Marker `<a-marker url="assets/pattern/NAME_DEINES_PATTERNS.patt"></a-marker>`

How To: Audio ändern

Voraussetzungen

Erstelle eine Audiodatei, welche die Alte ersetzen soll. Erstelle dazu eine WebVTT (https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API/Web_Video_Text_Tracks_Format)-Datei, welche die Untertitel dazu beinhaltet.

> ℹ Als Audioformat ist **MP3** zu empfehlen, da es von einer Mehrzahl an Browsern unterstützt wird.

Schritte

Audio ändern

1. Kopiere oder verschiebe die Audio- und Untertiteldatei in den Ordner **campusTour/audios**.

   > ⚠ Wenn du nur ein bestimmtes Audio ändern willst, dann ist es sinnvoll die Namen deiner neuen Dateien zu dem der alten, zu ändernden Dateien anzupassen und die alten Dateien somit einfach zu ersetzen. Damit werden die folgenden Schritte überflüssig.

2. Suche in der **ar.html**-Datei nach dem Marker, bei dem du das Audio ersetzen willst.

3. Ändere im `<source/>` - und im `<track>`-Tag die Pfade zu den Dateien entsprechend ab und passe, wenn nötig den Typ der `<source/>` an:

```
<audio>
    <source src="audios/NAME_DER_AUDIODATEI.mp3" type="audio/mpeg"/>
    <track src="audios/NAME_DER_UNTERTITELDATEI.vtt" default
```

```
kind="captions" label="Deutsch" srclang="de">
</audio>
```

> Wenn du ein weiteres Format zum Abspielen bereitstellen willst, dann kannst du auch einfach ein weiteren `<source/>`-Tag unter oder über den vorhandenen `<source/>`-Tag machen.

How To: Modell und Marker hinzufügen

Voraussetzungen

Wichtig sind die Voraussetzungen der vorherigen How-Tos (Modell ("Voraussetzungen" in "How-Tos: Einfache Änderungen"), Marker ("Voraussetzungen" in "How-Tos: Einfache Änderungen"), Audio ("Voraussetzungen" in "How-Tos: Einfache Änderungen")).

Schritte

Modell und Marker hinzufügen

Lade das Model von der Webseite herunter oder exportiere es aus dem Programm.

Lade von dem Marker Generator sowohl den Marker als auch das Bild herunter.

Erstelle eine Audio-Datei sowie eine Untertiteldatei (WebVTT-Format).

1. Kopiere oder verschiebe den Marker in den Ordner **campusTour/assets/patterns** und das Bild in den Ordner **campusTour/assets/images/**, das Modell mit allen Dateien in den Ordner **campusTour/models/** und die Audio- und Untertiteldatei in den Ordner **campusTour/audios**.

2. Erstelle in der `<a-scene>` ein neuen `<a-marker>`:

```
<a-scene>
    <a-maker
            preset="custom"
            type="pattern"
            size="0.25"
            emitevents="true"
            audiomarker <!-- Ist wichtig, damit der Marker auch
```

```
Audio abspielen kann, wenn er gefunden wird -->
    id="KENNZEICHNUNG_DES_MARKERS" <!-- Nicht mehr unbedingt
notwendig -->
    url="assets/patterns/NAME_DEINES_PATTERNS.patt"
    >
    <a-entity
            gltf-model="#MODELL_NAME"
            scale="0 0 0.1"
            position="0 0 0"
            rotation="0 0 0"
    ></a-entity>
    <audio>
        <source src="audios/NAME_DER_AUDIODATEI.mp3"
type="audio/mpeg"/>
        <track src="audios/NAME_DER_UNTERTITELDATEI.vtt" default
kind="captions" label="Deutsch" srclang="de">
    </audio>
    </a-maker>
</a-scene>
```

3. Ersetze alle Platzhalter (in CAPSLOCK geschrieben) im Code oben mit den entsprechenden Werten.

Referenz

HTML

<a-scene> (A-Frame Scene)

`<a-scene>` ist das Wurzelelement, das alle A-Frame-Elemente enthält. Es ist das Hauptelement, das die gesamte Szene definiert. Es ist ähnlich wie `<body>` in HTML, aber für AR.

> In diesem Projekt wird `<a-scene>` verwendet, um die AR-Szene zu definieren. In diesem Element sind alle AR-Elemente enthalten, wie z.B. `<a-entity>` und `<a-marker>`.

```
<a-scene>
    <a-box position="0 0.5 -5" rotation="0 45 0" color="#4CC3D9"></a-box>
</a-scene>
```

Siehe auch

A-Frame (https://aframe.io/docs/0.9.0/introduction/)

Parameter

- `embedded`: Definiert, ob die Szene in einem iframe eingebettet ist.
- `vr-mode-ui`: Definiert, ob die VR-Modus-Schaltfläche angezeigt wird.
- `arjs`: Definiert die AR.js Einstellungen.
- `renderer`: Definiert die Renderer-Einstellungen.

<a-entity> (A-Frame Entity)

`<a-entity>` ist ein generisches Element, das ein Objekt in der Szene darstellt. Es kann verwendet werden, um 3D-Objekte, Lichter, Kameras, Sound, Partikel und mehr hinzuzufügen.

> In diesem Projekt wird `<a-entity>` verwendet, um 3D-Objekte in der AR-Szene zu platzieren.

```
<a-entity geometry="primitive: box" material="color: blue" position="0
1.5 -5"></a-entity>
```

Siehe auch

A-Frame (https://aframe.io/docs/0.9.0/introduction/)

Parameter

- `gltf-model`: URL oder Referenz zum 3D-Modell.
- `position`: Position des Objekts in der Szene.
- `rotation`: Rotation des Objekts in der Szene.
- `scale`: Skalierung des Objekts in der Szene.
- `camera`: Definiert die Kamera in der Scene.

<a-marker> (AR.js Marker)

`<a-marker>` ist ein Element, das verwendet wird, um AR.js-Marker in der Szene zu definieren. Es wird in Kombination mit `<a-entity>` verwendet, um 3D-Objekte anzuzeigen, wenn der Marker erkannt wird.

> In diesem Projekt wird `<a-marker>` verwendet, um AR.js-Marker zu definieren und 3D-Objekte anzuzeigen, wenn der Marker erkannt wird.

```
<a-marker preset="hiro">
    <a-box position="0 0.5 0" rotation="0 45 0" color="#4CC3D9"></a-box>
</a-marker>
```

Siehe auch

AR.js (https://ar-js-org.github.io/AR.js-Docs)

Parameter

- `preset`: Voreingestellter Marker (hiro, kanji, barcode, custom).
- `type`: Typ des Markers (pattern, barcode, unknown).
- `url`: URL des benutzerdefinierten Markers.
- `size`: Größe des Markers.
- `emitevents`: Definiert, ob Ereignisse emittiert werden sollen.
- `audiomarker`: Definiert, dass der Marker Audio enthält, was beim Erkennen abgespielt werden soll.

<audio>

`<audio>` ist ein HTML-Element, das verwendet wird, um Audio in eine Webseite einzubinden. Hier wird es verwendet, um das Audio für die Modelle zu definieren.

In diesem Projekt wird `<audio>` verwendet, um Audiodateien für die Modelle zu definieren.

```
<audio>
    <source src="audios/first_message.mp3" type="audio/mpeg">
    <track src="audios/first_message.vtt" label="Deutsch" srclang="de"
kind="captions">
</audio>
```

Siehe auch

audio HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio) | HTML audio W3Schools (https://www.w3schools.com/tags/tag_audio.asp)

Parameter

- `src`: URL der Audiodatei.
- `type`: Typ der Audiodatei (audio/mpeg, audio/ogg, audio/wav).
- `preload`: Definiert, ob die Audiodatei beim Laden der Seite geladen werden soll.
- `autoplay`: Definiert, ob die Audiodatei automatisch abgespielt werden soll.
- `loop`: Definiert, ob die Audiodatei in einer Schleife wiedergegeben werden soll.
- `controls`: Definiert, ob die Audiodatei-Steuerung angezeigt werden soll.

<track>

`<track>` ist ein HTML-Element, das verwendet wird, um Untertitel oder Untertitelspuren für Audio- oder Video-Elemente zu definieren.

> In diesem Projekt wird `<track>` verwendet, um Untertitel für die Audiodateien zu definieren.

```
<track src="audios/first_message.vtt" label="Deutsch" srclang="de"
kind="captions">
```

Siehe auch

track HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/track) | HTML track W3Schools (https://www.w3schools.com/tags/tag_track.asp)

Parameter

- `src`: URL der Untertitelspur.
- `label`: Label der Untertitelspur.
- `srclang`: Sprache der Untertitelspur.
- `kind`: Art der Untertitelspur (subtitles, captions, descriptions, chapters, metadata).

`<div>` ist ein generisches Container-Element, das verwendet wird, um andere HTML-Elemente zu gruppieren.

> In diesem Projekt wird `<div>` verwendet, um Schaltflächen und Eingabefelder zu gruppieren.

```
<div style="position: absolute; top: 0; left: 0; right: 0; z-index: 1;">
    <button onclick="toggleFullscreen()">Fullscreen</button>
</div>
```

Siehe auch

div HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div) | HTML div W3Schools (https://www.w3schools.com/tags/tag_div.asp)

Parameter

- `id`: Eindeutige ID des Elements.
- `style`: CSS-Stile für das Element.

<button>

`<button>` ist ein HTML-Element, das verwendet wird, um eine Schaltfläche in eine Webseite einzubinden.

> In diesem Projekt wird `<button>` verwendet, um das Popup für das Ende anzuzeigen und zu bestätigen sowie für die erste Navigation zur AR-Szene.

```
<button onclick="toggleFullscreen()">Fullscreen</button>
```

Siehe auch

button HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button) | HTML button W3Schools (https://www.w3schools.com/tags/tag_button.asp)

Parameter

- `class`: CSS-Klasse des Elements.
- `onclick`: JavaScript-Funktion, die aufgerufen wird, wenn die Schaltfläche geklickt wird.

<label>

`<label>` ist ein HTML-Element, das verwendet wird, um ein Label für ein anderes Element zu definieren.

> In diesem Projekt wird `<label>` verwendet, um ein Label für ein Eingabefeld zu definieren.

```
<label for="name">Name:</label>
<input type="text" id="name">
```

Siehe auch

label HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label) | HTML label W3Schools (https://www.w3schools.com/tags/tag_label.asp)

Parameter

- `for`: ID des zugehörigen Input ("<input>" in "Referenz")-Elements.

<input>

`<input>` ist ein HTML-Element, das verwendet wird, um Benutzereingaben zu erfassen.

> In diesem Projekt wird `<input>` verwendet, um das Eingabefeld für den finalen Satz zu definieren.

```
<input type="text" id="name">
```

Siehe auch

input HTML Docs MDN (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) | HTML input W3Schools (https://www.w3schools.com/tags/tag_input.asp)

Parameter

- `type`: Typ des Eingabefelds (text, number, email, password, date, file).
- `id`: Eindeutige ID des Elements.
- `value`: Standardwert des Eingabefelds.
- `placeholder`: Platzhaltertext im Eingabefeld.
- `required`: Definiert, ob das Eingabefeld erforderlich ist.

CSS

CSS (Cascading Style Sheets) wird verwendet, um das Aussehen und Layout von HTML-Elementen zu definieren. Es ermöglicht die Trennung von Inhalt und Design und bietet eine Vielzahl von Stileigenschaften, um Webseiten zu gestalten. Hierbei wird der Syntax von CSS verwendet.

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}
```

Der Syntax besteht aus einem *Selektor* und einer oder mehreren Deklarationen, die durch geschweifte Klammern `{}` eingeschlossen sind. Jede Deklaration besteht aus einem *Eigenschaftsnamen* und einem *Wert*, die durch einen Doppelpunkt `:`.

Selektoren

Es gibt verschiedene Arten von Selektoren in CSS, die verwendet werden können, um HTML-Elemente zu stylen. Mit diesen wählt man die HTML-Elemente aus, denen man neue Stileigenschaften zuweisen möchte.

Beispiele von Selektoren sind:

```
p {
    color: red;
}

#header {
    font-size: 24px;
}

.button {
    background-color: blue;
}

img[src] {
    border: 1px solid black;
}

a:hover {
    color: green;
}
```

Siehe auch

CSS Selektoren MDN (https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors) | CSS Selektoren W3Schools (https://www.w3schools.com/css/css_selectors.asp)

JavaScript

JavaScript ist eine Programmiersprache, die verwendet wird, um Webseiten interaktiv zu gestalten. Es ermöglicht das Hinzufügen von Funktionen, Ereignissen und Animationen zu HTML-Elementen. Hierbei wird der Syntax von JavaScript verwendet.

Siehe auch

JavaScript MDN (https://developer.mozilla.org/en-US/docs/Web/JavaScript) | JavaScript W3Schools (https://www.w3schools.com/js/default.asp)

pxToXx

Die Funktionen `pxToVh(px)` und `pxToVw(px)` sind kleine Hilfsfunktionen, die verwendet werden, um Pixel in viewport height und width Werte umzurechnen. Damit können dann einfacher Responsive Designs erstellt werden.

```
function pxToVh(px) {
    return px / document.documentElement.clientHeight * 100;
}

function pxToVw(px) {
    return px / document.documentElement.clientWidth * 100;
}
```

valueOfUnitString

Diese Funktion extrahiert aus einer Zeichenkette mit einer Einheit (z.B. "10px", "50%") den numerischen Wert.

```
function valueOfUnitString(unitString) {
    const unitStringRegex = /([0-9.]+)([a-z%]+)/i;
    return parseFloat(unitStringRegex.exec(unitString)[1]);
}
```

isColliding

Diese Funktion überprüft, ob ein Bild sich mit einem schon platziertem Bild überschneiden würde, wenn es an platziert werden würde.

```
/**
 * @param placedImages - Ein Array von Bildern, die schon platziert
wurden
 * @param image - Das Bild, das platziert werden soll
 * @returns {boolean} - Gibt zurück, ob das Bild sich mit einem schon
platzierten Bild überschneiden würde
 */
function isColliding(placedImages, image) {
    return placedImages.some(img => Math.abs(img.top - image.top) <
img.height + image.height && Math.abs(img.left - image.left) < img.width
+ image.width);
}
```

Popup Funktionen

Diese Funktionen sind für das Anzeigen, das Schließen und das Validieren des Popups zuständig.

showPopup

Diese Funktion zeigt das Popup an.

```
function showPopup() {
    document.getElementById('popup').style.display = 'flex';
    document.getElementById('showPopupButton').style.display = 'none';
    document.addEventListener('click', closePopupOnClickOutside);
}
```

closePopup

Diese Funktion schließt das Popup.

```
function closePopup() {
    document.getElementById('popup').style.display = 'none';
    document.getElementById('showPopupButton').style.display = 'block';
    document.removeEventListener('click', closePopupOnClickOutside);
}
```

closePopupOnClickOutside

Diese Funktion schließt das Popup, wenn der Benutzer außerhalb des Popups klickt.

```
/**
 * @param {Event} event - Das click Event.
 */
function closePopupOnClickOutside(event) {
    const popup = document.getElementById('popup');
    const button = document.getElementById('showPopupButton');

    if (!popup.contains(event.target) && !button.contains(event.target))
{
        closePopup();
    }
}
```

checkSentence

Die Funktion überprüft, ob der eingeben Satz korrekt ist und, wenn der Satz korrekt ist, zeigt das Model an und lässt das Audio dazu abspielen.

```
function checkSentence() {
    const inputField = document.getElementById('endSentenceInput');
    const input = inputField.value;
    const correctSentence = /^Die Wahrheit liegt im Licht\.?$/i; //
Ersetze dies durch den korrekten Satz

    if (correctSentence.test(input)) {
        closePopup();
```

```
        const model = document.querySelector('a-entity#final');
        const audio = model.querySelector("audio");

        model.setAttribute('visible', true);

        audio.play().then(() => console.info(`Playing
${audio.currentSrc}`)).catch(e => console.error(e));
        audio.onended = () => {
            model.setAttribute('visible', false);
        }
    } else {
        setTimeout(() => alert(`Der Satz "${input}" ist falsch. Bitte
versuchen Sie es erneut.`), 500);
    }
    inputField.value = "";
}
```

Lade Funktion der Seite

Um Code auszuführen, wenn die Seite geladen wird, muss man den `window.onload` Event Listener verwenden.

```
window.onload = function () {
    getCleanUrl();
    setupInputFieldListener();
    setupAudioTracks();
    placeDecorationImages();
};
```

See also

External documentation

https://developer.mozilla.org/en-US/docs/Web/CSS (https://developer.mozilla.org/en-US/docs/Web/CSS)
https://developer.mozilla.org/en-US/docs/Web/JavaScript (https://developer.mozilla.org/en-US/docs/Web/JavaScript)
https://aframe.io/docs/0.9.0/introduction/ (https://aframe.io/docs/0.9.0/introduction/)

https://developer.mozilla.org/en-US/docs/Web/HTML (https://developer.mozilla.org/en-US/docs/Web/HTML)
https://ar-js-org.github.io/AR.js-Docs/ (https://ar-js-org.github.io/AR.js-Docs/)