AnimationMixer

Der AnimationMixer ist eine zentrale Klasse von Three.js zur Verwaltung von Animationen innerhalb einer 3D-Szene. Er übernimmt die zeitliche Steuerung, Wiedergabe und Kombination von Animationsabläufen. Ein AnimationMixer kann mehrere Animationen gleichzeitig verwalten und erlaubt beispielsweise das Starten, Stoppen oder Überblenden zwischen Bewegungen.

Im Kontext der Workshop-Pipeline wird der AnimationMixer relevant, wenn externe Animationsclips (z. B. glTF- oder FBX-Animationen) verwendet werden sollen. Im Workshop werden Bewegungen überwiegend prozedural per JavaScript-Code umgesetzt und daher noch ohne AnimationMixer realisiert.

Typisches Nutzungsschema:

```
const mixer = new THREE.AnimationMixer(model);
const action = mixer.clipAction(animationClip);
action.play();
```

Animationsclips

Animationsclips sind gespeicherte Bewegungssequenzen eines 3D-Modells. Sie enthalten zeitlich definierte Änderungen von Eigenschaften wie Position, Rotation, Skalierung oder Blendshape-Werten. Ein Animationsclip kann beispielsweise enthalten:

- Gehbewegungen
- Sprunganimationen
- Tanzsequenzen
- Gesichtsausdrücke
- Handgesten

In Three.js werden Animationsclips typischerweise gemeinsam mit einem AnimationMixer abgespielt.

Im Workshop werden Animationen zunächst als Bone-Rotationen direkt in JavaScript implementiert. Dies besitzt den architektonischen Vorteil, das die Bewegungsbeschreibung von der Anwendungslogik getrennt ist und so besser nachvollzieh- und anpassbar ist. Für komplexere Bewegungen, wie etwa natürliche Lauf- oder Tanzanimationen, werden in professionellen Pipelines in der Regel vorbereitete Animationsclips genutzt.

Avatar

Ein Avatar ist die digitale Repräsentation eines Benutzers, einer Figur oder einer künstlichen Entität innerhalb einer virtuellen Umgebung. Avatare können unterschiedliche Rollen übernehmen, etwa als persönliche Identität, Spielfigur, virtueller Assistent oder digitaler Performer.

Im Workshop bezeichnet der Begriff konkret ein humanoides 3D-Modell, das mit VRoid Studio erzeugt, als VRM-Datei exportiert und anschließend im Browser dargestellt wird.

Ein Avatar umfasst typischerweise mehrere Ebenen:

- geometrisches Modell
- Texturen und Materialien
- Skelettstruktur (Rig)
- Gesichtsanimationen
- Metadaten
- Interaktionslogik

In weiterführenden Konzepten, wie etwa Virtual-Idol- oder Metaverse-Systemen, kann ein Avatar zusätzlich mit Dialogsystemen, KI-Komponenten, Stimme und autonomen Verhaltensmechanismen erweitert werden.

Avaturn

Avaturn ist ein Werkzeug zur Erstellung realistischer 3D-Avatare. Die Plattform erlaubt die Generierung personalisierter Figuren, häufig auf Basis von Fotos oder parametrischen Anpassungen. Im Unterschied zu VRoid Studio, das stark auf stilisierte Anime- bzw. Manga-Charaktere fokussiert ist, verfolgt Avaturn einen semi-realistischen bis realistischen Charakterstil.

Im Workshop wird Avaturn als mögliche Alternative zu VRoid Studio genannt. Da Avaturn Avatare unter anderem als VRM-kompatible Modelle exportieren kann, lässt sich eine ähnliche Browser-Pipeline mit Three.js und @pixiv/three-vrm grundsätzlich ebenfalls aufbauen.

Typische Unterschiede zu VRoid Studio:

VRoid Studio	**Avaturn**
Anime-Stil	realistischer Stil
parametrische Charaktergestaltung	foto-/template-orientierte Erzeugung
direkte VRM-Orientierung	mehrere Exportoptionen

Blendshapes

Blendshapes sind ein Verfahren zur Verformung von 3D-Geometrie. Dabei werden alternative Zustände eines Modells definiert und interpoliert. Ein Blendshape kann beispielsweise darstellen:

- geöffneter Mund
- Lächeln
- geschlossene Augen
- gehobene Augenbrauen

Während der Laufzeit können diese Zielzustände mit unterschiedlichen Gewichtungen gemischt werden.

In VRM-Avataren spielen Blendshapes eine zentrale Rolle für:

- Facial Expressions

- LipSync
- Viseme-Animationen
- emotionale Darstellung

Im Workshop wird dies insbesondere relevant, wenn der Avatar mittels Text-to-Speech spricht. Dabei werden Blendshape-Werte genutzt, um Mundbewegungen zu simulieren.

Bone-Rotation

Eine Bone-Rotation beschreibt die Drehung eines einzelnen Knochens innerhalb eines Rig-Systems. Humanoide 3D-Modelle bestehen intern aus einer hierarchischen Knochenstruktur. Jeder Knochen besitzt eine Position und eine Orientierung relativ zu seinem Elternknoten. Typische Bones sind beispielsweise:

- Head
- Chest
- LeftUpperArm
- RightLowerLeg
- Hips

Im Workshop werden Animationen direkt über Bone-Rotationen erzeugt:

```
rightUpperArm.rotation.z += 0.5;
```

Dadurch lassen sich Gesten, Bewegungen und Körperhaltungen programmatisch erzeugen.

Bone-Rotationen bilden das technische Fundament vieler Animationstechniken:

- prozedurale Animation
- Keyframe-Animation
- Motion-Capture-Retargeting
- inverse Kinematik

CORS (Cross-Origin Resource Sharing)

CORS ist ein Sicherheitsmechanismus moderner Webbrowser, der den Zugriff auf Ressourcen unterschiedlicher Ursprünge (Origins) reguliert. Eine Origin wird im Wesentlichen bestimmt durch:

- Protokoll
- Hostname
- Port

Browser verhindern standardmäßig viele Zugriffe zwischen unterschiedlichen Origins. Im Workshop wird CORS relevant, wenn:

- VRM-Dateien geladen werden,
- externe Texturen eingebunden werden,
- lokale Dateien direkt über `file://` geöffnet werden.

Aus diesem Grund wird im Workshop ein lokaler Webserver verwendet.

Typischer Fehlerfall:

```
Access to fetch blocked by CORS policy
```

Die Nutzung eines lokalen Servers, z. B. über VS Code Live Server, Node.js oder Python HTTP Server, umgeht diese Problematik.

CSS (Cascading Style Sheets)

CSS ist die Standardtechnologie zur Gestaltung von Webseiten. Während HTML die Struktur einer Anwendung definiert, übernimmt CSS deren visuelle Darstellung. Typische CSS-Aufgaben sind:

- Farben
- Layout
- Abstände
- Größen
- Responsive Design
- Animationen
- Benutzeroberflächen-Design

Im Workshop wird CSS genutzt, um:

- das Bedienpanel zu gestalten,
- Buttons zu formatieren,
- Dialogfenster zu definieren,
- Overlay-Elemente über dem 3D-Canvas anzuordnen.

```
button {
    border-radius: 10px;
    background: white;
}
```

Damit bildet CSS einen durchaus relevanten Bestandteil der webbasierten Laufzeitumgebung neben HTML, JavaScript und Three.js.

Facial Expressions

Facial Expressions bezeichnen die Darstellung von Gesichtsausdrücken eines digitalen Avatars. Sie dienen dazu, Emotionen, Reaktionen oder kommunikative Nuancen visuell darzustellen. Typische Facial Expressions sind beispielsweise:

- Lächeln
- Traurigkeit
- Überraschung

- Wut
- Blinzeln
- Nachdenklichkeit

Technisch werden Facial Expressions bei vielen Avatar-Systemen über Blendshapes realisiert. Dabei werden vorbereitete Gesichtsformen mit unterschiedlichen Gewichtungen aktiviert.

In VRM-Avataren existieren häufig bereits vordefinierte Gesichtsausdrücke.

Im Workshop wird das Thema insbesondere relevant für:

- sprechende Avatare
- LipSync
- emotionale Dialogsysteme
- Virtual-Idol-Inszenierungen

Eine einfache Steuerung kann beispielsweise erfolgen über:

```
expressionManager.setValue("happy", 1.0);
```

glTF / glTF 2.0

glTF (GL Transmission Format) ist ein offenes Dateiformat zur effizienten Speicherung und Übertragung von 3D-Inhalten. Es wurde von der Khronos Group entwickelt und gilt heute als eines der wichtigsten Austauschformate für Echtzeit-3D-Anwendungen. Ein glTF-Modell kann unter anderem enthalten:

- Geometrie
- Materialien
- Texturen
- Animationen
- Kameras
- Lichtquellen
- Szenenstruktur

glTF 2.0 ist die aktuell verbreitete Hauptversion.

Im Workshop besitzt glTF eine besondere Bedeutung, da VRM technisch auf glTF 2.0 basiert.

VRM kann daher vereinfacht verstanden werden als:

glTF 2.0
+ humanoides Rig
+ Blendshapes
+ Avatar-Metadaten
+ VRM-spezifische Erweiterungen

Three.js lädt glTF-Dateien typischerweise über den `GLTFLoader`.

HTML (HyperText Markup Language)

HTML ist die Standard-Auszeichnungssprache des World Wide Web. HTML definiert die Struktur einer Webanwendung. Typische HTML-Elemente sind:

- Überschriften
- Formulare
- Buttons
- Eingabefelder
- Container
- Dialoge
- Canvas-Elemente

Im Workshop bildet HTML die Grundlage der Browser-Anwendung. Es definiert beispielsweise:

- Upload-Buttons für VRM-Dateien
- Bedienleisten
- Texteingaben für Sprache und ELIZA
- Dialogfenster für Stimmenauswahl
- das Canvas zur Darstellung der 3D-Szene

Beispiel:

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

Das Canvas-Element dient als Zeichenfläche für Three.js und damit für die Darstellung des Avatars.

HTML übernimmt somit die strukturelle Organisation der gesamten Benutzungsoberfläche.

Humanoid-Rig

Ein Humanoid-Rig ist eine standardisierte Skelettstruktur für humanoide 3D-Figuren. Es beschreibt, welche Knochen ein Modell besitzt und wie diese hierarchisch organisiert sind. Typische Elemente eines Humanoid-Rigs sind:

- Hips
- Spine
- Chest
- Head
- UpperArm
- LowerArm
- Hand
- UpperLeg
- LowerLeg
- Foot

Die Verwendung eines standardisierten Rigs erleichtert:

- Animationen
- Wiederverwendbarkeit

- Motion-Capture-Integration
- Austausch zwischen Plattformen

VRM definiert ein eigenes humanoides Rig-Schema.

Dadurch kann Code plattformunabhängig auf den gewünschten Knochen zugreifen:

```
vrm.humanoid.getNormalizedBoneNode("rightUpperArm")
```

Im Workshop ist das Humanoid-Rig die Grundlage sämtlicher Bewegungsanimationen.

JavaScript

JavaScript ist die zentrale Programmiersprache moderner Webanwendungen. Sie wird direkt im Browser ausgeführt und übernimmt die dynamische Steuerung von Benutzeroberflächen, Datenverarbeitung und Interaktionen.

Im Workshop fungiert JavaScript als ausführende Logikebene der gesamten Pipeline. Es steuert unter anderem:

- Three.js-Initialisierung
- Laden von VRM-Dateien
- Avatar-Animationen
- Kameralogik bzw. Avatar-Transformationen
- Text-to-Speech
- ELIZA-Dialogsystem
- Hintergrund- und Szenensteuerung

Ein typisches Beispiel:

```
button.addEventListener("click", () => {
    speak(textInput.value);
});
```

JavaScript bildet damit das Bindeglied zwischen:

HTML →	Struktur
CSS →	Gestaltung
Three.js	3D-Rendering
VRM	Avatargestaltung

Ohne JavaScript wäre die Pipeline lediglich eine statische Webseite.

LookAt-Verhalten

Das LookAt-Verhalten beschreibt die Fähigkeit eines Avatars, seinen Blick dynamisch auf ein Zielobjekt auszurichten. Dabei werden typischerweise Kopf-, Augen- oder Halsbewegungen angepasst:

- Avatar schaut zur Kamera
- Avatar verfolgt den Mauszeiger
- Avatar blickt einen Gesprächspartner an
- Avatar richtet Aufmerksamkeit auf Objekte

VRM unterstützt LookAt-Mechanismen standardmäßig. Die Bibliothek @pixiv/three-vrm stellt dafür Funktionen bereit. Ein vereinfachtes Konzept lautet z. B. :

Zielpunkt bestimmen
→ Blickrichtung berechnen
→ Head-/Eye-Rotation anpassen

LookAt-Verhalten trägt wesentlich zur sozialen Glaubwürdigkeit virtueller Figuren bei.

Im Workshop kann dies perspektivisch erweitert werden, etwa durch:

- Blick zur sprechenden Person
- Blickverhalten im Dialogsystem
- kamerageführte Aufmerksamkeit
- * Virtual-Idol-Performance-Szenarien

Node.js

Node.js ist eine JavaScript-Laufzeitumgebung außerhalb des Webbrowsers. Während JavaScript ursprünglich primär innerhalb von Browsern ausgeführt wurde, ermöglicht Node.js die Nutzung derselben Sprache auch für:

- Serveranwendungen
- Entwicklungswerkzeuge
- Build-Prozesse
- lokale Entwicklungsumgebungen

Im Workshop wird Node.js vor allem als Werkzeug der Entwicklungsumgebung relevant. Typische Einsatzgebiete sind hier:

- lokaler Webserver
- Paketverwaltung über npm
- Projektinitialisierung
- Entwicklungswerkzeuge

Ein einfacher lokaler Server kann beispielsweise über Node-basierte Werkzeuge bereitgestellt werden. Node.js ist damit kein Bestandteil des Avatars selbst, sondern unterstützt die technische Infrastruktur der Pipeline.

npm (Node Package Manager)

npm ist der Standard-Paketmanager von Node.js. Er dient der Installation, Aktualisierung und Verwaltung externer Softwarebibliotheken. In modernen Webprojekten wird npm verwendet, um Abhängigkeiten komfortabel einzubinden. Beispiele dafür sind:

- Three.js
- Vite
- Express
- TypeScript
- React

Ein typischer npm-Befehle ist: (hier die Installation des Three-Pakets)

```
npm install three
```

Im Workshop wird die Anwendung bewusst zunächst CDN-basiert aufgebaut, um die Einstiegshürde niedrig zu halten. In weiterführenden Projekten kann npm jedoch eine professionellere Projektorganisation ermöglichen.

npx

npx ist ein Werkzeug zum direkten Ausführen von Node-Paketen. Im Unterschied zu npm muss eine Anwendung dabei nicht dauerhaft global installiert werden.

Beispiel:

```
npx vite
```

Startet das Werkzeug `Vite`, ohne dass dieses vorher global eingerichtet werden muss.

npx wird häufig verwendet für:

- Projektgeneratoren
- Entwicklungsserver
- Build-Werkzeuge
- einmalige Hilfsprogramme

In erweiterten Varianten der Workshop-Pipeline kann npx hilfreich sein, wenn eine modernere Entwicklungsumgebung aufgebaut wird.

Phoneme

Phoneme sind die kleinsten bedeutungsunterscheidenden Laute einer Sprache. Beispiele im Deutschen:

- a
- e

- m
- p
- sch

Beim Sprechen entsteht Sprache aus zeitlich aufeinanderfolgenden Phonemen.

In Avatar-Systemen sind Phoneme relevant für:

- Sprachsynthese
- LipSync
- Mundanimation

Da 3D-Modelle keine akustischen Laute darstellen können, werden Phoneme häufig in Viseme übersetzt.

Phonem
→ sprachlicher Laut
Viseme
→ visuelle Mundform

Im Workshop wird dies relevant, wenn Text-to-Speech mit sprechenden Avataren kombiniert wird.

Polygonmodellierung

Polygonmodellierung ist eine grundlegende Technik der 3D-Computergrafik. Dabei werden Objekte aus polygonalen Flächen aufgebaut. Die häufigsten Polygonformen sind:

- Dreiecke
- Vierecke (Quads)

Ein 3D-Avatar besteht letztlich aus vielen miteinander verbundenen Polygonen. Diese definieren:

- Körperform
- Gesicht
- Kleidung
- Haare
- Accessoires

In Werkzeugen wie VRoid Studio erfolgt die Polygonmodellierung weitgehend parametrisch und benutzungsfreundlich. Der Benutzer arbeitet primär über Regler und visuelle Anpassungen statt über manuelle Mesh-Bearbeitung. Polygonmodellierung bildet die geometrische Grundlage jedes VRM-Avatars.

Regelbasierter Chatbot

Ein regelbasierter Chatbot erzeugt Dialoge auf Basis vordefinierter Regeln und Mustererkennung. Im Gegensatz zu modernen generativen KI-Systemen verwendet er keine großen Sprachmodelle. Ein typischer Ablauf lautet:

Eingabe analysieren
→ Muster erkennen
→ passende Regel auswählen
→ Antwort erzeugen

Beispiel:

User: "I am sad."
*Regel: "I am **"*
Antwort: "Why are you sad?"

Im Workshop wird dieses Konzept mit Hilfe von ELIZA demonstriert.

Render-Loop

Der Render-Loop ist die kontinuierliche Aktualisierungsschleife einer Echtzeit-3D-Anwendung. In jeder Iteration werden typischerweise folgende Schritte durchgeführt:

Zeit aktualisieren
→ Animationen berechnen
→ Szenenzustand aktualisieren
→ Bild rendern
→ nächsten Frame anfordern

In Three.js wird dies meist mit:

```
requestAnimationFrame()
```

umgesetzt.

Beispiel:

```
function animate() {
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
}
```

Im Workshop übernimmt der Render-Loop unter anderem:

- Aktualisierung des Avatars
- Animationen
- LookAt-Verhalten
- VRM-Updates
- Szenendarstellung

Der Render-Loop bildet damit das zeitliche Herzstück der Anwendung.

Rigging

Rigging bezeichnet den Prozess der Ausstattung eines 3D-Modells mit einer steuerbaren Skelettstruktur. Dabei werden Knochen definiert und mit der Geometrie des Modells verbunden. Ein Rig erlaubt anschließend:

- Bewegungen
- Animationen
- Körperhaltungen
- Gesichtssteuerung

Der typische Ablauf lautet:

3D-Modell
→ Skelett erzeugen
→ Bones verbinden
→ Gewichtung definieren
→ animierbarer Charakter

In VRoid Studio wird Rigging automatisch erzeugt. Dadurch erhalten exportierte VRM-Avatare bereits ein funktionsfähiges humanoides Rig.

Im Workshop muss daher kein vollständiges manuelles Rigging durchgeführt werden, um dennoch dessen Ergebnis bei Animationen und Bone-Rotationen nutzen zu können.

TTS (Text-to-Speech)

Text-to-Speech (TTS) bezeichnet Verfahren zur automatischen Umwandlung von geschriebenem Text in synthetische Sprache. Ein TTS-System verarbeitet typischerweise:

Texteingabe
→ linguistische Analyse
→ Lauterzeugung
→ Audiosignal

TTS-Systeme können unterschiedliche Qualitätsstufen besitzen, von einfachen synthetischen Stimmen bis hin zu nahezu natürlich klingenden KI-generierten Stimmen.

Im Workshop wird TTS genutzt, um den Avatar sprechen zu lassen. Die Textausgabe stammt beispielsweise aus:

- freier Texteingabe
- ELIZA-Dialogen
- später möglichen KI-Dialogsystemen

Die eigentliche Sprachsynthese erfolgt in der Browser-Pipeline über die WebSpeech API.

Three.js

Three.js ist eine weit verbreitete JavaScript-Bibliothek zur Entwicklung von 3D-Webanwendungen. Sie vereinfacht den Zugriff auf WebGL erheblich und abstrahiert viele technische Details moderner 3D-Grafikprogrammierung. Three.js stellt unter anderem bereit:

- Szenenverwaltung
- Kameramodelle
- Beleuchtung
- Materialien
- Geometrien
- Animation
- Asset-Loader

Ein typischer Three.js-Grundaufbau besteht aus:

Scene
Camera
Renderer
Render-Loop

Beispiel:

```
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera();
const renderer = new THREE.WebGLRenderer();
```

Im Workshop übernimmt Three.js die Rolle der zentralen Rendering-Engine. Es steuert:

- 3D-Szene
- Avatar-Darstellung
- Beleuchtung
- Kamera
- Texturen
- Boden- und Wandflächen
- Echtzeitaktualisierung

Three.js / three-vrm

Die Kombination Three.js / three-vrm bezeichnet die Zusammenarbeit zwischen der allgemeinen 3D-Rendering-Bibliothek Three.js und der VRM-spezifischen Erweiterung three-vrm. Während Three.js generische 3D-Objekte verarbeiten kann, ergänzt three-vrm Funktionen für humanoide Avatar-Modelle. Die Kombination ermöglicht:

VRM laden
→ Avatar interpretieren
→ Humanoid-Rig nutzen

→ Expressions steuern

→ LookAt-Verhalten ausführen

Diese Kombination bildet das technische Kernstück der gewählten Workshop-Pipeline.

@pixiv/three-vrm

@pixiv/three-vrm ist eine spezialisierte Bibliothek zur Nutzung von VRM-Avataren innerhalb von Three.js. Die Bibliothek wurde ursprünglich von Pixiv entwickelt. Sie erweitert Three.js um VRM-spezifische Funktionen wie:

- VRM-Import
- Humanoid-Zugriffe
- Expressions
- LookAt-System
- Metadatenverarbeitung
- Avatar-Updates

Typischer Import:

```
import { VRM } from "@pixiv/three-vrm";
```

Im Workshop ist @pixiv/three-vrm eine zentrale Kernbibliothek. Sie stellt die technische Brücke zwischen:

VRM-Datei ↔ Three.js-Szene

dar. Ohne diese Bibliothek müsste ein erheblicher Teil der VRM-spezifischen Logik selbst implementiert werden.

Viseme

Ein Visem beschreibt die visuelle Darstellung eines Sprachlautes. Während ein Phonem einen akustischen Laut repräsentiert, beschreibt ein Visem dessen sichtbare Mundform, zum Beispiel:

→ A weit geöffneter Mund |
→ O gerundete Lippen |
→ M geschlossene Lippen |

Avatar-Systeme verwenden Viseme zur Erzeugung von Lippensynchronität (LipSync). Typische Umsetzung:

Text
→ Phoneme
→ Viseme
→ Blendshape-Steuerung

Im Workshop wird dieses Konzept perspektivisch relevant, wenn Sprachsynthese und Gesichtsanimation enger gekoppelt werden.

VRM

VRM ist ein standardisiertes Dateiformat für humanoide 3D-Avatare. Technisch basiert VRM auf glTF 2.0, erweitert dieses jedoch um avatarbezogene Eigenschaften. VRM ergänzt glTF unter anderem:

- Humanoid-Rig
- Blendshapes
- Expressions
- LookAt-Konfiguration
- Metadaten
- Nutzungsinformationen

VRM = glTF 2.0 + Avatar-Erweiterungen

Im Workshop bildet VRM das zentrale Austauschformat zwischen:

VRoid Studio
→ Browser-Anwendung
→ Three.js / three-vrm

Die Standardisierung von VRM erleichtert die Interoperabilität zwischen unterschiedlichen Werkzeugen und Plattformen.

VRoid Studio

VRoid Studio ist ein Werkzeug zur Erstellung humanoider 3D-Avatare. Der Schwerpunkt liegt auf einer benutzungsfreundlichen, visuell geführten Charaktergestaltung. So können ohne tiefgehende 3D-Modellierungskenntnisse Avatare erzeugt werden. VRoid Studio erlaubt unter anderem:

- Gesichtsdesign
- Körperanpassung
- Frisuren
- Kleidung
- Farben
- Materialparameter

Die Avatare können anschließend direkt als VRM-Dateien exportiert werden.

Im Workshop bildet VRoid Studio den Einstiegspunkt der Pipeline.

VS Code (Visual Studio Code)

VS Code ist eine weit verbreitete Entwicklungsumgebung für Software- und Webentwicklung. Sie unterstützt zahlreiche Programmiersprachen und Erweiterungen.

Im Workshop wird VS Code vorgeschlagen für:

- HTML-Bearbeitung
- CSS-Bearbeitung
- JavaScript-Entwicklung
- Projektorganisation
- lokales Testen

Hilfreiche Erweiterungen sind beispielsweise:

- Live Server
- Syntax Highlighting
- JavaScript-Tools
- Git-Integration

Webserver

Ein Webserver stellt Dateien und Anwendungen über HTTP bzw. HTTPS bereit. Browserbasierte Anwendungen werden typischerweise nicht direkt aus dem Dateisystem (`file://`) geladen, sondern über einen Server ausgeliefert.

Im Workshop ist ein lokaler Webserver wichtig für:

- VRM-Dateiladen
- Modulimporte
- CORS-Vermeidung
- Web-API-Kompatibilität

Mögliche Varianten:

- VS Code Live Server
- Node.js-Server
- Python HTTP Server

Der Browser greift anschließend typischerweise auf diesen lokalen Webserver zu, indem eine bestimmt Port-Numer gewählt wird:

```
http://localhost:8000
```

WebSpeech API

Die WebSpeech API ist eine Browser-Schnittstelle für Sprachverarbeitung. Sie umfasst zwei Hauptbereiche:

- Speech Synthesis
- Speech Recognition

Die Workshop-Pipeline nutzt primär die Speech Synthesis API. Sie ermöglicht:

- Stimmenauswahl
- Sprachsynthese
- Spracheinstellungen
- Sprechtempo
- Tonhöhenkontrolle

```
const utterance = new SpeechSynthesisUtterance("Hallo Welt");
speechSynthesis.speak(utterance);
```

Die WebSpeech API bildet im Workshop die Grundlage für:

- sprechende Avatare
- ELIZA-Antwortausgabe
- experimentelle Sprachschnittstellen

Perspektivisch könnte zusätzlich auch Speech Recognition integriert werden, um Spracheingaben statt Texteingaben zu ermöglichen.