Dokumentation
Website: Das Sonnensystem

Name: Yannis Bär **Kurs: E2** **Datum: 01.06.2026**

Inhaltsverzeichnis

1. Projektidee und Ziel
2. Ordnerstruktur und Dateiaufbau
3. Entwicklungsablauf der Hauptseite
4. Unterseite „Planeten“
5. Unterseite „Ratespiel“
6. Unterseite „Quellen“
7. Eigene Arbeit und genutzte Hilfen
8. Zeitplan

1. Projektidee und Ziel

Mein Ziel war eine interaktive Website zum Sonnensystem erstellen, die nicht nur schlicht Informationen zeigt, sondern wie eine „kleine Reise“ durch das Weltall wirkt. Deshalb habe ich mich für einen dunklen Sternenhimmel, große Planetendarstellungen und klickbare Infoboxen entschieden. Die Hauptseite sollte beim Scrollen gewissermaßen das Gefühl geben, durch den Weltraum zu fliegen. Zusätzlich sollten die Unterseiten jeweils eine eigene Funktion haben: eine Übersicht über alle Planeten, ein Ratespiel mit Eingabefeld und eine Quellen-Seite.

2. Ordnerstruktur und Dateiaufbau

Die Dateien habe ich in einem Projektordner abgelegt, damit Bilder, HTML-Dateien und CSS-Dateien klar getrennt bleiben.

```
website/
├── index.html
├── planeten.html
├── ratespiel.html
├── quellen.html
├── style.css
├── planeten.css
├── ratespiel.css
├── quellen.css
├── Dokumentation.pdf
└── img/
    ├── merkur.png
    ├── venus.png
    ├── erde.png
    ├── mars.png
    ├── jupiter.png
    ├── saturn.png
    ├── uranus.png
    ├── neptun.png
    └── voyager.png
```

3. Entwicklungsablauf der Hauptseite

Die Hauptseite war der wichtigste Teil des Projekts. Ich habe zuerst die Grundstruktur in HTML geschrieben und die Bereiche in einzelne Szenen unterteilt. Jede Szene steht für einen Abschnitt im Sonnensystem. Dadurch kann man durch Scrollen von „Planet zu Planet reisen“. Im Folgenden werde ich hauptsächlich den JavaScript-Code erläutern, da dieser mithilfe von KI erstellt wurde und hauptsächlich für die index.hmtl Hauptseite, aber auch teilweise für die Unterseiten genutzt wird.

3.1 Grundaufbau mit Canvas, Scroll-Bereich und Container

Der Sternenhimmel im Hintergrund wurde mit einem <canvas>-Element umgesetzt. Das Canvas liegt mit position: fixed über die gesamte Bildschirmfläche und bleibt damit immer sichtbar. Damit die Seite trotzdem scrollbar ist, obwohl der eigentliche Inhalt fest liegt, habe ich zusätzlich ein unsichtbares Element mit der Höhe des gesamten Projekts verwendet.

Im Skript wird die Höhe des Scroll-Bereichs anhand der Anzahl der Szenen berechnet. Der eigentliche Inhalt liegt in #world und wird per transform: translate3d(...) nach oben verschoben. Dadurch bleibt der Canvas stehen, während die Szene darunter bewegt wird. Den nötigen JavaScript-Code am Ende des index.html Dokuments habe ich mithilfe ChatGPT erstellt.

3.2 Der Sternenhimmel

Das Skript generiert ein Array aus 6.000 zufällig verteilten Sternen mit unterschiedlichen Eigenschaften (Größe, Helligkeit, Tiefenwirkung). Beim Scrollen wird die Bewegung der Sterne verändert, sodass durch unterschiedliche Geschwindigkeiten ein gewisser räumlicher 3D-Effekt (ein sogenannter Parallax-Effekt) entsteht.

```
const STAR_COUNT = 6000;
const EXTRA_END_SPACE = 1.8;
```

```
const STAR_COUNT = 6000;
const EXTRA_END_SPACE = 1.8;


  for (let i = 0; i < STAR_COUNT; i++) {
    stars.push({
      x: Math.random() * w,
      y: Math.random() * totalHeight,
      size: rand(0.5, 2.6),
      depth: rand(0.25, 1),
      alpha: rand(0.15, 1),
      glow: Math.random() < 0.18 ? rand(2, 6) : 0
    });
  }
}
```

Im draw()-Loop werden die Sterne mit ctx.arc gezeichnet. Die y-Position wird mit currentScroll multipliziert, damit sich die Sterne im Verhältnis zum Scrollen verschieben. Zusätzlich wurde bei manchen Sternen ein kleiner Glow ergänzt. In der Animationsschleife wird die Position jedes Sterns mit seiner Tiefe verrechnet: const y = s.y - currentScroll * s.depth;. Sterne im "Hintergrund" (geringe Tiefe) bewegen sich langsamer als Sterne im "Vordergrund".

```
    function draw() {
      ctx.clearRect(0, 0, w, h);
      ctx.fillStyle = "#000";
      ctx.fillRect(0, 0, w, h);

      targetScroll = clamp(window.scrollY, 0, maxScroll);
      currentScroll += (targetScroll - currentScroll) * 0.08;

      world.style.transform = `translate3d(0, ${-currentScroll}px, 0)`;
      updateProgress();
```

Die Bewegung beim Scrollen stoppt nicht abrupt, sondern gleitet sanft nach (currentScroll += (targetScroll - currentScroll) * 0.08). Zudem werden Sterne, die gerade außerhalb des sichtbaren Bildschirms liegen (if (y < -80 || y > h + 80) continue;), gar nicht erst gezeichnet.

3.3 Navigation und sanftes Scrollen

Die obere Menüleiste dient als schnelle Navigation zu den Abschnitten Merkur bis Voyager. Wenn man auf einen Menüpunkt klickt, wird ein eigener Smooth-Scroll gestartet. Dabei wird die Zielposition berechnet und dann mit einer eigenen Ease-Funktion animiert.

```
    menuLinks.forEach(link => {
      link.addEventListener("click", (e) => {
        const href = link.getAttribute("href");
        if (!href || !href.startsWith("#")) return;

        e.preventDefault();

        const target = document.querySelector(href);
        if (!target) return;

        const offset = 70;
        const y = Math.max(0, target.offsetTop - offset);

        history.pushState(null, "", href);
        smoothScrollTo(y, 1900);
      });
    });
```

Die Funktion smoothScrollTo verwendet eine sogenannte „Cubic-Ease-Funktion“. Am Anfang bewegt sich die Seite daher langsam, in der Mitte schneller und am Ende wieder ruhiger.

3.4 Klickbare Planeten und Infoboxen

Jeder Planet ist als anklickbares Element aufgebaut. Der Planet selbst besteht aus einem Link mit Bild und Label. Beim Klick öffnet sich rechts oder links eine Infobox mit Fakten und einer kurzen Beschreibung. Ob die Infobox sichtbar ist, wird über die Klasse active gesteuert.

```
    function openPlanet(scene) {
      if (activeScene && activeScene !== scene) {
        closePlanet(activeScene);
      }

      activeScene = scene;
      scene.classList.add("active");

      const link = scene.querySelector(".planet-link");
      const info = scene.querySelector(".planet-info");

      if (link) link.classList.add("active");
      if (info) info.setAttribute("aria-hidden", "false");
    }
```

Die Infobox ist zuerst unsichtbar. In CSS hat sie opacity: 0 und pointer-events: none. Sobald die Szene aktiv wird, blendet sie mit opacity: 1 ein und wird anklickbar. Gleichzeitig verschwindet die Beschriftung des Planeten und das Bild wird vergrößert.

```
.planet-info {
  width: var(--info-w, min(38vw, 430px));
  max-width: 100%;
  max-height: min(88vh, 920px);
  overflow: auto;
  padding: var(--info-pad, 26px);
  border-radius: 34px;
  background: rgba(10, 10, 20, 0.58);
  backdrop-filter: blur(14px);
  border: 1px solid rgba(255, 255, 255, 0.11);
  box-shadow: 0 18px 50px rgba(0, 0, 0, 0.38);
  opacity: 0;
  transform: translateX(0) scale(0.96);
  pointer-events: none;
  transition: opacity 0.45s ease, transform 0.45s ease, box-shadow 0.45s ease;
  position: relative;
  min-width: 0;
```

Die Infoboxen haben unterschiedliche Breiten und Abstände, weil ich die Werte mit CSS-Variablen direkt pro Abschnitt setze. So kann Jupiter viel größer erscheinen als Merkur. Das ist sinnvoll, weil die Planeten im Projekt nicht alle gleich groß wirken sollen.

3.5 Fortschrittsanzeige links

Links auf der Seite gibt es eine kleine Fortschrittsanzeige mit einem leuchtenden Kreis. Sie zeigt, wie weit man „im Sonnensystem“ bzw. auf der Website schon gescrollt hat.

```
function updateProgress() {
  const progress = maxScroll > 0 ? clamp(currentScroll / maxScroll, 0, 1) : 0;
  progressRail.style.setProperty("--progress", progress.toString());
  progressThumb.style.top = `${progress * 100}%`;
}
```

Der Kreis wandert also proportional zum aktuellen Scroll-Stand nach unten.

3.6 Menüleiste

Die Menüleiste ist die zentrale und permanente Leiste am oberen Bildschirmrand. Sie bleibt beim Scrollen durch eine feste Platzierung immer im Vordergrund sichtbar. Die Menüleiste besitzt ein horizontales Layout. Die Planeten-Links werden exakt in der Mitte zentriert, Links zu Unterseiten rechtsbündig ausgerichtet. Beim Hovern (siehe Codebeispiel) vergrößern und erhellen sich die Navigationselemente über eine zeitliche Verzögerung.

```
.menu a {
  color: #fff;
  text-decoration: none;
  padding: 8px 14px;
  border-radius: 999px;
  background: rgba(255, 255, 255, 0.08);
  transition: transform 0.25s ease, background 0.25s ease;
  flex: 0 0 auto;}

.menu a:hover {
  transform: scale(1.12);
  background: rgba(255, 255, 255, 0.18);
}
```

3.7 Infoboxen für die Planeten

Die Infoboxen enthalten die Detaildaten der Planeten und blenden sich erst bei Interaktion ein. (Design wurde teilweise von KI überarbeitet (CSS)).

JavaScript vergibt bei Klick eine Aktiv-Klasse, woraufhin das CSS die Box von 0 auf 100% Sichtbarkeit erhöht und leicht hochskaliert. Im unsichtbaren Zustand sind die Boxen für Mausereignisse komplett gesperrt, um keine Elemente im Hintergrund zu blockieren. Die Boxen nutzen außerdem CSS-Variablen aus dem HTML, um ihre Breite an die wirkliche Größe des jeweiligen Planeten anzupassen.

4. Unterseite „Planeten“

Die Planeten-Seite zeigt alle Planeten auf einen Blick und ist nicht scrollbar. Der JavaScript-Code der Hauptseite wurde in Teilen auf die Unterseiten übernommen. Der Hintergrund ist deshalb wieder ein Sternenhimmel mit Canvas, damit die Unterseite optisch zur Hauptseite passt. Die Planeten schweben auf einer leichten Bogenform statt in einer geraden Linie.

Jeder Planet bekommt im HTML eigene Werte für Position, Verzögerung und Größe. Dadurch kann ich Merkur klein und Jupiter groß darstellen. Beim Laden werden die Planeten nacheinander eingeblendet.

Das style.css Dokument wurde ebenfalls in Teilen in ein neues CSS-Dokument übernommen und ergänzt, damit es übersichtlicher ist und den Inhalt der Unterseite spezifisch abdeckt, die Formatierung und das grundlegende Design bleibt jedoch einheitlich.

5. Unterseite „Ratespiel“

Das Ratespiel ist ein interaktives Quiz mit 10 zufälligen Fragen. Die Seite hat drei Zustände: Startbildschirm, Quiz und Endbildschirm. Der Spieler startet bewusst über den Button „Spiel starten“. Erst dann wird die erste Frage geladen. Das JavaScript-Dokument für das Spiel wurde mithilfe von KI erstellt. Der restliche Inhalt von Zeile 1-91 ist vollständig selbst geschrieben.

```
function startGame() {
  score = 0;
  currentIndex = 0;
  questions = shuffle(QUESTION_POOL).slice(0, 10);

  showScreen(quizScreen);
  renderQuestion();
}
```

Die Fragen kommen aus einer großen QUESTION_POOL-Liste. Mit shuffle() wird diese Liste gemischt und dann werden nur 10 Einträge genommen. Dadurch ist jede Runde anders.

```
function checkAnswer(rawValue) {
  const q = questions[currentIndex];
  const userValue = Number(rawValue);

  if (userValue === q.a) {
    score += 1;
  }
}
```

Die Eingabe wird als Zahl gelesen und direkt mit der erwarteten Antwort verglichen. Weil nur Zahlen (const userValue = Number(rawValue)) gefragt sind, ist die Logik bewusst einfach gehalten.

Der Endbildschirm zeigt die Gesamtpunktzahl an, wodurch das Spiel einen klaren Abschluss hat.

6. Unterseite „Quellen“

Die Quellen-Seite dient zur Aufführung von externen Links, Bildquellen und Hinweise zur verwendeten Literatur. Zusätzlich enthält sie eine Tabelle, in der die wichtigsten Kategorien übersichtlich aufgelistet werden. Einen Download-Button (siehe Codebeispiel) für die Dokumentation habe ich ebenfalls eingebaut.

```
<a href="Dokumentation.pdf" download class="download-btn">
  Dokumentation herunterladen
</a>
```

7. Eigene Arbeit und genutzte Hilfen

Ein sehr großer Teil der Webseite und des Codes wurde von mir selbst erstellt und anschließend schrittweise verbessert. Lediglich der JavaScript-Code und Teile des CSS-Codes wurden mithilfe von KI erstellt. Die Prompts sind leider zu umfangreich, um sie direkt anzugeben, da ich mich schrittweise der gewünschten Lösung genähert habe und mir gleichzeitig den Code habe erklären lassen. KI wurde von mir meistens im Dialog verwendet, weshalb ich keine einfachen großen Prompts angeben kann, die die Gesamtlösung ergeben.

W3Schools (https://www.w3schools.com/html/) habe ich als Nachschlagewerk benutzt, wenn ich mir bei der Syntax unsicher war oder eine Funktion noch einmal nachlesen wollte. KI habe ich vor allem genutzt, um mir der Erstellung des JavaScript-Codes für den Hintergrund der Hauptseite und das Ratespiel helfen zu lassen.

8. Zeitplan

Der folgende Zeitplan zeigt den Ablauf, wie ich das Projekt schrittweise innerhalb von drei Wochen umgesetzt habe.

Phase	Inhalt
1. Planung	Idee, Farben, Aufbau, Ordnerstruktur
2. Hauptseite	Canvas, Scrollraum, Szenen, Sterne
3. Details	Infoboxen, Hover, Navigation, Fortschritt
4. Planeten-Seite	Anordnung, Einblendung
5. Ratespiel	Fragenpool, Zufall, Antwortprüfung, Endscreen
6. Quellen-Seite	Links, Tabelle, Download-Button
7. Überarbeitung	Layout, Lesbarkeit, Testen auf Fehler