Anleitung zur Datenanzeige und Seitenübergängen

Verstanden! Ich werde die Anleitung entsprechend den korrekten Funktionen der Seiten aktualisieren:

---

1. Worum geht’s?

Diese Anleitung erklärt, wie du die bereitgestellten HTML- und JavaScript-Dateien nutzen kannst, um dynamische TV-Slides für einen Fernseher draußen zu erstellen. Die Slides werden auf dem Bildschirm angezeigt und wechseln automatisch zwischen verschiedenen Inhalten.

Hier erfährst du:

- **Wie du die Slides dynamisch aktualisieren kannst:** Die Seiten zeigen verschiedene Informationen und wechseln mit coolen Übergängen.
- **Wie du Daten hinzufügen kannst:** Du kannst Daten leicht anpassen oder neue hinzufügen, um den Inhalt der Slides zu ändern.
- **Die `fetchData`-Methode als Platzhalter:** Die `fetchData`-Methode ist nur ein Beispiel, wie du Daten einbinden kannst. Du musst diese Methode durch deinen eigenen Code ersetzen, um die tatsächlichen Daten zu laden und anzuzeigen.

Im Grunde geht es darum, die TV-Slides so einzurichten, dass du automatisch aktualisieren und optisch ansprechend auf dem Bildschirm präsentiert werden.

---

2. Projektübersicht

Das Projekt umfasst mehrere HTML-Dateien (`index.html`, `recap.html`, `post.html`, `sendungenImDetail.html`, `veranstaltungen.html`, `social.html`), die Daten anzeigen und mithilfe von JavaScript für Weiterleitungen und Animationen sanft zwischen ihnen wechseln.

3. Dateistrukturübersicht

```
/projekt-verzeichnis
|-- index.html
|-- recap.html
|-- post.html
|-- sendungenImDetail.html
|-- veranstaltungen.html
|-- social.html
|-- css/
|   |-- style.css
|-- img/
|   |-- greybg.png
```

```
|   |-- father_wink_high_heller.png
|   |-- daughter_phone3_high_heller.png
|   |-- generic-image-placeholder.png
|-- js/
|   |-- sendungenImDetail.js
|   |-- recap.js
|-- data/
|   |-- sendedata.csv
```

4. Verstehen der HTML-Struktur

4.1 `index.html`

Diese Seite dient als **Willkommensseite** und leitet nach 5 Sekunden automatisch zu `recap.html` weiter:

- **Automatische Weiterleitung**: Weiterleitung nach `recap.html` nach 5 Sekunden.

```
setTimeout(function() {
  window.location.href = 'recap.html';
}, 5000);
```

4.2 `recap.html`

Diese Seite bietet eine Zusammenfassung der Sendungszahlen:

- **Zähler**: Dynamisch aktualisierene Zähler für neue Kunden und Sendungen.
- **Top-Kunden**: Zeigt Bilder der Top-Kunden.
- **Ladeanimation**: Eine Throbber-Animation wird angezeigt, bis die Daten geladen sind.
- **Weiterleitung**: Leitet nach 7 Sekunden automatisch zu `post.html` weiter.

```
document.addEventListener("DOMContentLoaded", function() {
  document.querySelectorAll('.fade-in').forEach(element => {
    element.classList.add('visible');
  });
});

setTimeout(function() {
  window.location.href = 'post.html';
}, 7000);
```

4.3 `post.html`

Diese Seite dient als **Vorlage für allgemeine Posts oder Nachrichten** und leitet nach einer Verzögerung sanft zu `sendungenImDetail.html` weiter:

- **Hintergrundbild**: Zeigt ein zentriertes Bild mit Text und einem Fade-In-Effekt.
- **Weiterleitung**: Leitet nach 5 Sekunden automatisch weiter.

```
document.addEventListener("DOMContentLoaded", function() {
  document.body.style.opacity = 1;
  setTimeout(function() {
    window.location.href = 'sendungenImDetail.html';
  }, 5000);
});
```

4.4 `sendungenImDetail.html`

Zeigt Details zu Sendungen an, einschließlich eines Diagramms und der Sendungszahlen, und leitet zu `veranstaltungen.html` weiter:

- **Text und Diagramm**: Kombiniert Textüberschriften mit einem Diagramm für Sendungszahlen.
- **Fade-In-Effekt**: Wird auf Bilder und Text angewendet.
- **Weiterleitung**: Automatische Weiterleitung nach 7 Sekunden.

```
document.addEventListener("DOMContentLoaded", () => {
  document.querySelectorAll('.fade-in').forEach(element => {
    element.classList.add('visible');
  });
});

setTimeout(function() {
  window.location.href = 'veranstaltungen.html';
}, 7000);
```

4.5 `veranstaltungen.html`

Diese Seite zeigt kommende Veranstaltungen an und leitet nach 7 Sekunden automatisch zu `social.html` weiter:

- **Veranstaltungsliste**: Zeigt mehrere Veranstaltungseinträge mit Tagen, Titeln und Agenden.
- **Weiterleitung**: Automatische Weiterleitung nach 7 Sekunden.

```
document.addEventListener("DOMContentLoaded", () => {
  document.querySelectorAll('.chart-long').forEach(element => {
    element.classList.add('visible');
  });
});

setTimeout(function() {
```

```
  window.location.href = 'social.html';
}, 7000);
```

4.6 `social.html`

Diese Seite zeigt einen sozialthematischen Hintergrund und leitet nach 5 Sekunden zurück zu `index.html`:

- **Hintergrundbild**: Zeigt ein LinkedIn-Themenbild mit einem Fade-In-Effekt.
- **Weiterleitung**: Leitet nach 5 Sekunden zurück zu `index.html`.

```
document.addEventListener("DOMContentLoaded", function() {
  document.body.style.opacity = 1;
  setTimeout(function() {
    window.location.href = 'index.html';
  }, 5000);
});
```

5. Einbinden von Daten auf der Website

Hier erfährst du, wie du die Daten aus `sendedata.csv` einbinden kannst, um die Slides zu aktualisieren.

5.1 Daten laden und verarbeiten

- **CSV-Daten laden** (in `recap.js`):
  - Die Daten werden aus einer CSV-Datei (`sendedata.csv`) gelesen und geparst.
  - Diese Daten werden verwendet, um die Anzahl der Sendungen auf der Zusammenfassungsseite zu aktualisieren.

  Beispiel:

```
async function readSendeData() {
  const csvData = localStorage.getItem("sendedata.csv");
  const records = [];
  if (!csvData) {
    console.error("CSV-Daten nicht im Local Storage gefunden");
    return records;
  }

  const lines = csvData.split("\n").map(line => line.trim()).filter(line =>
line.length > 0);
  const headers = lines[0].split(",");

  for (let i = 1; i < lines.length; i++) {
    const values = lines[i].split(",");
    try {
```

```
      const record = new CsvRecord(
        parseInt(values[headers.indexOf("Jahr")], 10),
        parseInt(values[headers.indexOf("Monat")], 10),
        parseInt(values[headers.indexOf("Elektronisch")], 10),
        parseInt(values[headers.indexOf("Physisch")], 10)
      );
      records.push(record);
    } catch (error) {
      console.error(`Datenformatfehler in Zeile ${i}: ${error.message}`);
    }
  }
  return records;
}
```

- **Daten aktualisieren**:
  - Nachdem die Daten geladen und geparst wurden, wird die Benutzeroberfläche dynamisch aktualisieren.
  - Die `updateNumbers`-Funktion aktualisieren die Anzahl der elektronischen und physischen Sendungen:

```
function updateNumbers(records) {
  if (records.length === 0) {
    console.error("Keine Datensätze verfügbar, um die Zahlen zu aktualisieren");
    return;
  }
  const latestRecord = records[records.length - 1];

  document.getElementById("elektrSendungen").textContent =
latestRecord.electronic.toString();
  document.getElementById("sendungen").textContent =
latestRecord.physical.toString();
}
```

- **Diagramm aktualisieren** (in `sendungenImDetail.js`):
  - Die `getPercentAndColor`-Funktion berechnet die Prozentsätze der verschiedenen Sendungstypen und aktualisieren das Kreisdiagramm entsprechend:

```
let postalischperc =
Number((document.getElementById("postalischevalue").innerText.replace('.',
'').replace(',', '')) /
Number(document.getElementById("gesamtvalue").innerText.replace('.',
'').replace(',', ''))) * 100;
let elektronischeperc =
Number((document.getElementById("elektronischevalue").innerText.replace('.',
'').replace(',', '')) /
```

```
Number(document.getElementById("gesamtvalue").innerText.replace('.',
'').replace(',', ''))) * 100;
let regmailSendungenperc =
Number((document.getElementById("regamilvalue").innerText.replace('.',
'').replace(',', '')) /
Number(document.getElementById("gesamtvalue").innerText.replace('.',
'').replace(',', ''))) * 100;

var ctx = document.getElementById('pie').getContext("2d");
var myPieChart = new Chart(ctx, {
  type: 'pie',
  data: {
    labels: ['Postalisch', 'Elektronisch', 'Regmail'],
    datasets: [{
      data: [postalischperc, elektronischeperc, regmailSendungenperc],
      backgroundColor:
```

[
"rgba(20, 20, 20, 0.2)",
"rgba(0, 128, 255, 0.2)",
"rgba(255, 0, 0, 0.2)"
],
borderColor: [
"rgba(20, 20, 20, 1)",
"rgba(0, 128, 255, 1)",
"rgba(255, 0, 0, 1)"
],
borderWidth: 1
}]
},
options: {
responsive: true,
maintainAspectRatio: false,
plugins: {
legend: {
position: 'top',
},
tooltip: {
callbacks: {
label: function(tooltipItem) {
return tooltipItem.label + ': ' + tooltipItem.raw.toFixed(2) + '%';
}
}

```
}
}
}
});
```

6. Handhabung von sanften Übergängen und Animationen

6.1 Fade-In-Effekte

Jede Seite verwendet einen Fade-In-Effekt für sanftere Übergänge. Dies wird durch die Einstellung der `opacity`-Eigenschaft und die Verwendung von CSS-Übergängen erreicht:

```
body {
  opacity: 0; /* Zunächst ausgeblendet */
  transition: opacity 2s ease-in-out; /* Passen du die Dauer für den Fade-In-Effekt an
*/
}
```

6.2 Weiterleitungen mit Verzögerung

JavaScript steuert das Timing der Übergänge zwischen Seiten mithilfe von `setTimeout`. Passen du die Verzögerungsdauer in Millisekunden nach Bedarf an:

```
setTimeout(function() {
  window.location.href = 'nextpage.html';
}, 5000); // 5 Sekunden Verzögerung
```

6.3 Anpassen von Animationen

Ändern du die Dauer und das Timing der Übergänge, indem du die CSS-`transition`-Eigenschaften anpassen:

```
body {
  transition: opacity 3s ease-in-out; /* 3 Sekunden Fade-In */
}
```

7. Anpassung der Website

Du kannst die Website weiter anpassen, indem du die HTML-, CSS- und JavaScript-Dateien modifizierst:

- **Datenquellen aktualisieren:** verknüpfe eine API oder eine andere automatische Datenquelle.
- **Anzeigelogik anpassen:** Passen du an, wie Daten präsentiert werden oder wie Übergänge

funktionieren.

- **CSS-Anpassungen:** Ändere `styles.css`, um das Erscheinungsbild der Website zu ändern.
