Happy-Hour Update – Apps Script

Dieses Skript ermöglicht dir, Folien automatisch basierend auf Zeitfester ein- oder auszublenden. Hier ist, wie du es verwenden kannst:

1. Öffnen deine Google Slides Präsentation
2. Gehen zu Erweiterungen > Apps Script
3. Fügen den Code ein und ersetzen Sie 'DEINE\_PRÄSENTATIONS\_ID' mit der ID deiner Präsentation

Dieses neue Skript bietet verschiedene Möglichkeiten zur zeitbasierten Steuerung deiner Folien. Du kannst in den Notizen der Folien folgende Befehle verwenden:

1. Vordefinierte Zeitfenster:
#show-during: happy-hour
2. Benutzerdefinierte Zeiträume:
#show-time: 14:00-16:30
3. Mehrere Zeitfenster für eine Folie:
#show-times: 09:00-12:00, 14:00-16:00
4. Unter der Funktion "function addCustomTimeWindow()" im Skript kannst du neue Zeitfenster hinzufügen:
updateTimeWindow('coffee-break', '10:30', '11:00');

Die vordefinierten Zeitfenster sind:

* morning: 09:00-12:00
* lunch: 12:00-14:00
* afternoon: 14:00-17:00
* happy-hour: 17:00-19:00
* evening: 19:00-23:00

Du kannst diese Zeiten im Code anpassen oder neue Zeitfenster hinzufügen. Das Skript prüft alle 5 Minuten, ob Folien ein- oder ausgeblendet werden sollen.

**WICHTIG:**

Unter "Erweiterungen > Apps Script", musst du kein Dienst oder eine Bereitstellung angeben. Aber du musst ein Trigger erstellen, damit das Skript automatisch zu bestimmten Zeiten ausgeführt wird.

Sie musst du folgendes tun:

1. Klicke auf "Ausführen" (▶️) um das Skript einmal manuell zu testen
2. Nach dem ersten Test klicke auf das Uhr-Symbol "⏰" in der linken Seitenleiste (Trigger)
3. Klicke auf den blauen Button "+ Trigger hinzufügen"
4. Stelle die Trigger-Einstellungen wie folgt ein:
	1. Funktion: manageTimeBasedSlides
	2. Ereignisquelle: "Zeit"
	3. Zeitintervall: "Minuten-Timer"
	4. Intervall wählen Sie am besten 5 oder 10 Minuten

Das sorgt dafür, dass deine Präsentation automatisch alle 5-10 Minuten aktualisiert wird und die richtigen Folien basierend auf der Uhrzeit anzeigt oder ausblendet.

function manageTimeBasedSlides() {

  // ID deiner Präsentation

  const PRESENTATION\_ID = 'DEINE\_PRÄSENTATIONS\_ID';

  // Präsentation öffnen

  const presentation = SlidesApp.openById(PRESENTATION\_ID);

  // Aktuelle Zeit

  const now = new Date();

  const currentHour = now.getHours();

  const currentMinute = now.getMinutes();

  const currentTime = currentHour \* 60 + currentMinute; // Zeit in Minuten seit Mitternacht

  // Zeitfenster definieren (können Sie nach Bedarf anpassen)

  const timeWindows = {

    'morning': { start: '09:00', end: '12:00' },

    'lunch': { start: '12:00', end: '14:00' },

    'afternoon': { start: '14:00', end: '17:00' },

    'happy-hour': { start: '17:00', end: '19:00' },

    'evening': { start: '19:00', end: '23:00' }

  };

  // Hilfsfunktion zum Konvertieren von "HH:MM" in Minuten seit Mitternacht

  function timeToMinutes(timeString) {

    const [hours, minutes] = timeString.split(':').map(Number);

    return hours \* 60 + minutes;

  }

  // Hilfsfunktion zum Prüfen, ob aktuelle Zeit in einem Zeitfenster liegt

  function isInTimeWindow(windowStart, windowEnd) {

    const startMinutes = timeToMinutes(windowStart);

    const endMinutes = timeToMinutes(windowEnd);

    return currentTime >= startMinutes && currentTime < endMinutes;

  }

  // Alle Folien durchgehen

  const slides = presentation.getSlides();

  slides.forEach((slide, index) => {

    const notes = slide.getNotesPage().getSpeakerNotesShape().getText().asString();

    // Verschiedene Zeitsteuerungsbefehle überprüfen

    // 1. Spezifische Zeitfenster

    // Syntax: #show-during: happy-hour

    const showDuringMatch = notes.match(/#show-during:\s\*([\w-]+)/);

    if (showDuringMatch) {

      const timeWindow = timeWindows[showDuringMatch[1]];

      if (timeWindow) {

        slide.setSkipped(!isInTimeWindow(timeWindow.start, timeWindow.end));

      }

    }

    // 2. Benutzerdefinierte Zeiträume

    // Syntax: #show-time: 14:00-16:30

    const showTimeMatch = notes.match(/#show-time:\s\*(\d{2}:\d{2})-(\d{2}:\d{2})/);

    if (showTimeMatch) {

      const [\_, start, end] = showTimeMatch;

      slide.setSkipped(!isInTimeWindow(start, end));

    }

    // 3. Mehrere Zeitfenster

    // Syntax: #show-times: 09:00-12:00, 14:00-16:00

    const showTimesMatch = notes.match(/#show-times:\s\*((?:\d{2}:\d{2}-\d{2}:\d{2}(?:,\s\*)?)+)/);

    if (showTimesMatch) {

      const timeRanges = showTimesMatch[1].split(',').map(range => range.trim());

      let isVisible = false;

      for (const range of timeRanges) {

        const [start, end] = range.split('-');

        if (isInTimeWindow(start, end)) {

          isVisible = true;

          break;

        }

      }

      slide.setSkipped(!isVisible);

    }

  });

}

// Trigger erstellen für regelmäßige Überprüfung (z.B. alle 5 Minuten)

function createTimeBasedTrigger() {

  // Bestehende Trigger löschen

  const triggers = ScriptApp.getProjectTriggers();

  triggers.forEach(trigger => ScriptApp.deleteTrigger(trigger));

  // Neuen Trigger alle 5 Minuten erstellen

  ScriptApp.newTrigger('manageTimeBasedSlides')

    .timeBased()

    .everyMinutes(5)

    .create();

}

// Hilfsfunktion zum Hinzufügen oder Aktualisieren von benutzerdefinierten Zeitfenstern

function updateTimeWindow(name, startTime, endTime) {

  const userProperties = PropertiesService.getUserProperties();

  const timeWindows = JSON.parse(userProperties.getProperty('timeWindows') || '{}');

  timeWindows[name] = {

    start: startTime,

    end: endTime

  };

  userProperties.setProperty('timeWindows', JSON.stringify(timeWindows));

}

// Beispiel für das Hinzufügen eines neuen Zeitfensters

function addCustomTimeWindow() {

  // Beispiel: Neue "Coffee Break" Zeit hinzufügen

  updateTimeWindow('coffee-break', '10:30', '11:00');

}