1. הדמיית נתונים מצטברים: הדף כולל קטע (#aggregatedData) המציג נתוני טיסה מצטברים באמצעות תרשימים שונים. זה כולל ספירת טיסות לכל חברת תעופה, זמן עיכוב ממוצע לכל חברת תעופה, ספירת טיסות לפי סוג, עיכוב מקסימלי לכל חברת תעופה ומדינות שהוגשו לכל חברת תעופה. תכונה זו משתמשת ב-Plotly.js להמחשת הנתונים, מה שמקל על המשתמשים להבין דפוסי טיסה וסטטיסטיקות במבט חטוף.

```
function displayAggregatedData(aggregatedData):
```

קטע קוד זה עוסק בהצגת נתוני טיסה מצטברים באמצעות Plotly.js. הפונקציה displayAggregatedData(aggregatedData) אחראית ליצירת תרשימים שונים על סמך נתוני הטיסה שנצברו והועברו אליו. בואו נפרק את הפרטים:

מטרת הקוד

המטרה העיקרית של קוד זה היא להציג נתוני טיסה מצטברים בצורה ידידותית למשתמש. מטרתו היא לספק תובנות לגבי:

- מספר הטיסות לכל חברת תעופה.
- זמן העיכוב הממוצע לכל חברת תעופה לאורך זמן.
- התפלגות ספירת הטיסות לפי סוג (למשל, הגעה, יציאה).
- העיכוב המרבי שחווים טיסות של כל חברת תעופה.
- מספר המדינות המשרתות כל חברת תעופה.

איך זה עובד

הפונקציה displayAggregatedData(aggregatedData) לוקחת פרמטר אחד, aggregatedData, שצפוי להיות אובייקט המכיל נתוני טיסה מצטברים מראש. נתונים אלה כוללים ספירת טיסות לכל חברת תעופה, זמני עיכוב ממוצעים, ספירת טיסות לפי סוג, עיכובים מרביים ומדינות שהוגשו לכל חברת תעופה.

תצורת עלילה נפוצה

האובייקט commonPlotConfig מגדיר קבוצה של אפשרויות סגנון נפוצות עבור כל התרשימים כדי לשמור על מראה ותחושה עקביים. זה כולל צבעי רקע, צבע גופן וגודל גופן. אופרטור ההתפשטות (...) משמש למיזוג הגדרות נפוצות אלה עם תצורות תרשים בודדות.

זְרִימָה

1. הפונקציה מגדירה אובייקט commonPlotConfig המכיל אפשרויות תצורה משותפות לכל החלקות, כגון צבע הרקע, צבע הגופן וגודל הגופן.
2. הפונקציה משתמשת בשיטת Plotly.newPlot כדי ליצור כל חלקה. הוא מספק את הנתונים הדרושים ואפשרויות התצורה עבור כל חלקה.
3. הפונקציה משתמשת באופרטור התפשטות (...) כדי למזג את ה-commonPlotConfig עם אפשרויות התצורה הספציפיות עבור כל חלקה.
4. כל חלקה נוצרת עם מזהה ייחודי ומצורפת למסמך ה-HTML.
5. הפונקציה חוזרת על שלבים 2-4 עבור כל חלקה.

ספירת טיסות לכל חברת תעופה (תרשים עמודות)

תרשים זה משתמש בסוג הפס כדי לייצג את מספר הטיסות לכל חברת תעופה. ציר ה-x מייצג חברות תעופה, וציר ה-y מייצג את ספירת הטיסות. תבנית ריחוף מותאמת אישית משמשת להצגת מידע מפורט בעת ריחוף מעל פס.

זמן עיכוב ממוצע לכל חברת תעופה (תרשים קו)

תרשים זה משרטט את זמן ההשהיה הממוצע לכל חברת תעופה לאורך זמן תוך שימוש בתרשים פיזור עם מצב קווים+סמנים. הוא מדמיין כיצד זמן העיכוב הממוצע של כל חברת תעופה משתנה. מניחים שציר ה-x מייצג זמן, וציר ה-y מייצג את ההשהיה הממוצעת בדקות.

ספירת טיסות לפי סוג (תרשים עוגה)

תרשים עוגה זה מציג את התפלגות ספירת הטיסות לפי סוג (למשל, הגעה, יציאה). הוא משתמש בתוויות עבור סוגי טיסות ובערכים עבור ספירות. תבנית ריחוף מותאמת אישית מספקת מידע מפורט על ריחוף.

עיכוב מקסימלי לכל חברת תעופה (תרשים עמודות)

בדומה לתרשים העמודות הראשון, זה מייצג את העיכוב המרבי שחווים טיסות של כל חברת תעופה, עם עיכובים שהומרו מאלפיות שניות לשעות לקריאה. הצבע האדום משמש לטושים להדגשת עיכובים.

מדינות שהוגשו לפי חברת תעופה (תרשים עמודות)

תרשים זה מדגים את מספר המדינות המשרתות כל חברת תעופה, תוך שימוש באורך מערך המדינות עבור כל חברת תעופה. הצבע הירוק נבחר עבור סמנים, שכנראה מייצג היבט חיובי (גיוון ביעדי השירות).

עיצוב רספונסיבי

כל תרשים נעשה רספונסיבי באמצעות האפשרות { responsive: true }. זה מבטיח שהתרשימים מתאימים את גודלם בהתאם לגודל המיכל, מה שהופך את האפליקציה ליותר ידידותית למשתמש במכשירים שונים.

2. תצוגת שעון בזמן אמת:

```
updateClock(); // Update the clock immediately
setInterval(updateClock, 1000); // Update the clock every second
```

קטע הקוד אחראי לעדכון תצוגת השעון בדף אינטרנט בכל שנייה.

זרימה

1. הפונקציה `updateClock` נקראת מיד כדי לעדכן את תצוגת השעון.
2. הפונקציה `setInterval` משמשת כדי לקרוא לפונקציה `updateClock` כל שנייה, מה שמבטיח שתצוגת השעון מתעדכנת באופן רציף.

updateClock:

קטע הקוד הוא פונקציה בשם `updateClock` אשר מעדכנת את התוכן של רכיב HTML עם השעה הנוכחית.

זרימה

1. הפונקציה `updateClock` נקראת.
2. התאריך והשעה הנוכחיים מתקבלים באמצעות האובייקט `Date`.
3. השעות, הדקות והשניות נשלפות מהתאריך ומומרות למחרוזות.
4. הערכים שחולצו מרופדים באפסים מובילים במידת הצורך.
5. מחרוזת הזמן נבנית באמצעות השעות, הדקות והשניות המעוצבות.
6. התוכן של אלמנט ה-HTML עם המזהה "clock" מתעדכן במחרוזת הזמן.

3. **לחצן גלילה למעלה: כפתור "גלול לראש הדף" (#scrollTopBtn) מאפשר למשתמשים לנווט בקלות חזרה לראש העמוד לאחר הגלילה למטה. זה משפר את חווית המשתמש בדפים ארוכים.**

```
const scrollToTopHandler = () => window.scrollTo({ top: 0, behavior: 'smooth'
});
addEventListenerToElement(scrollTopBtn, 'click', scrollToTopHandler);
```

scrollToTopHandler:

קטע הקוד הוא פונקציה שגוללת את החלון לראש העמוד עם אנימציה חלקה.

זרימה

1. לאחר לחיצה על הכפתור, הפונקציה `scrollToTopHandler` נקראת.
2. הפונקציה משתמשת בשיטת `window.scrollTo` כדי לגלול את החלון למעלה.
3. הפרמטר 'למעלה' מוגדר ל-0 כדי לציין את החלק העליון של העמוד.
4. הפרמטר 'התנהגות' מוגדר ל'חלק' כדי לאפשר אנימציית גלילה חלקה.

4. סרגל ניווט דביק: סרגל הניווט (div.sticky-nav) נועד להיות דביק, כלומר הוא נשאר גלוי בחלק העליון של נקודת התצוגה כשהמשתמש גולל מטה בעמוד. זה מספק גישה קלה למסנני החיפוש ולפקדים אחרים מבלי צורך לגלול בחזרה למעלה.

```
.sticky-nav {
            position: sticky;
            top: 0;
            padding: 10px;
            z-index: 1000;
            background-color: rgba(51, 51, 51, 0.8);
            transition: opacity 0.3s ease;
        }
```

קטע הקוד מגדיר את סגנונות ה-CSS עבור סרגל ניווט דביק שנשאר קבוע בחלק העליון של הדף בעת הגלילה.

זרימה

קטע הקוד מגדיר מחלקת CSS הנקראת "sticky-nav". מחלקה זו מוחלת על רכיב סרגל הניווט. המאפיין " position: sticky " מבטיח שסרגל הניווט יישאר קבוע בראש העמוד בעת הגלילה. המאפיין "top: 0" ממקם את סרגל הניווט בחלק העליון של נקודת התצוגה. המאפיין "padding: 10px" מוסיף ריפוד לסרגל הניווט. המאפיין "z-index: 1000" מבטיח שסרגל הניווט יופיע מעל אלמנטים אחרים בדף. המאפיין "background-color: rgba(51, 51, 51, 0.8)" מגדיר את צבע הרקע של סרגל הניווט לאפור כהה שקוף למחצה. המאפיין "transition: opacity 0.3s ease" מוסיף אפקט מעבר חלק לאטימות של סרגל הניווט.

5. שינוי אטימות בגלילה: סרגל הניווט הדביק משנה את האטימות שלו בהתבסס על מיקום הגלילה. כאשר המשתמש גולל מטה, האטימות פוחתת, מה שהופך את סרגל הניווט לפחות בולט. ריחוף מעל סרגל הניווט מחזיר אותו לאטימות מלאה.

```
const stickyNav = document.querySelector('.sticky-nav'); // Select the sticky
navigation bar
        let scrollThreshold = 10; // threshold for when the user starts
scrolling down
        addEventListenerToElement(window, 'scroll', function () {
            if (window.scrollY > scrollThreshold) {
                stickyNav.classList.add('low');
            } else {
                stickyNav.classList.remove('low');
            }
        });

        // Add mouseenter event listener to remove 'low' class on hover
        addEventListenerToElement(stickyNav, 'mouseenter', function () {
            this.classList.remove('low');
        });

        // Add mouseleave event listener to re-add 'low' class when mouse
leaves
        // Only re-add 'low' if the page is scrolled down past the threshold
        addEventListenerToElement(stickyNav, 'mouseleave', function () {
            if (window.scrollY > scrollThreshold) {
                this.classList.add('low');
            }
        });
```

קטע הקוד אחראי על הוספה והסרה של מחלקת CSS ('נמוכה') לסרגל ניווט דביק על סמך התנהגות הגלילה ואינטראקציה של העכבר של המשתמש.

כניסות

- `stickyNav`: רכיב סרגל הניווט הדביק שנבחר באמצעות `document.querySelector('.sticky-nav')`.

- `scrollThreshold`: ערך הסף (במקרה זה 10) שקובע מתי המשתמש מתחיל לגלול מטה.

זרימה

1. הקוד בוחר את אלמנט סרגל הניווט הדביק באמצעות `document.querySelector('.sticky-nav')` ומקצה אותו למשתנה `stickyNav`.

2. הוא מגדיר משתנה `scrollThreshold` עם ערך של 10, המייצג את הסף למועד בו המשתמש מתחיל לגלול מטה.

3. הקוד מוסיף מאזין אירועי גלילה לאובייקט `window` באמצעות `addEventListenerToElement(window, 'scroll', function () {...})`.

4. כאשר המשתמש גולל, מאזין האירועים בודק אם ערך `window.scrollY` (מיקום גלילה אנכי) גדול מ-`scrollThreshold`.

5. אם הערך `window.scrollY` גדול יותר מ-`scrollThreshold`, הקוד מוסיף את מחלקת ה-'low' CSS לאלמנט `stickyNav` באמצעות `stickyNav.classList.add('low')`.

6. אם הערך `window.scrollY` אינו גדול מ-`scrollThreshold`, הקוד מסיר את מחלקת ה-'low' CSS מהאלמנט `stickyNav` באמצעות `stickyNav.classList.remove('low')`.

7. הקוד מוסיף גם מאזין אירועי mouseenter לאלמנט `stickyNav` באמצעות `addEventListenerToElement(stickyNav, 'mouseenter', function () {...})`.

8. כאשר המשתמש מרחף מעל האלמנט `stickyNav`, מאזין האירועים מסיר את מחלקת ה-'low' CSS מהאלמנט `stickyNav` באמצעות `this.classList.remove('low')`.

9. הקוד מוסיף מאוזן אירועים של mouseleeave לאלמנט `stickyNav` באמצעות `addEventListenerToElement(stickyNav, 'mouseleave', function () {...})`.

10. כאשר המשתמש מרחיק את העכבר מאלמנט `stickyNav`, מאזין האירועים בודק אם הערך `window.scrollY` גדול מה`scrollThreshold`.

11. אם הערך `window.scrollY` גדול מ-`scrollThreshold`, הקוד מוסיף את מחלקת ה-'low' CSS לאלמנט ה-stickyNav באמצעות `this.classList.add('low')`.

פלטים

- הקוד מוסיף את מחלקת ה-'low' CSS לאלמנט 'stickyNav' כאשר המשתמש גולל מטה מעבר ל-'scrollThreshold'.

- הקוד מסיר את מחלקת ה-CSS 'נמוכה' מהאלמנט 'stickyNav' כאשר המשתמש מרחף מעליו.

- הקוד מוסיף מחדש את מחלקת ה-'low' CSS לאלמנט 'stickyNav' כאשר המשתמש מרחיק ממנו את העכבר, אך רק אם הדף גולל מטה מעבר ל-'scrollThreshold'.

6.סינון דינמי ועדכון נתונים: פונקציונליות החיפוש לא רק מסננת את הטיסות הגלויות בטבלה אלא גם מעדכנת את תרשימי הנתונים המצטברים בהתבסס על התוצאות המסוננות. זה מבטיח שההדמיות משקפות תמיד את קריטריוני החיפוש הנוכחיים.

7. זיהוי חברת תעופה מאוחרת: תכונה לזיהוי והצגת מידע על חברת התעופה המאוחרת ביותר (#lateAirlineInfo). זה כולל חישוב לאיזו חברת תעופה יש הכי הרבה עיכובים והצגת מידע זה למשתמש, שיפור היכולות האנליטיות של הדף.

```
function findMostLateAirline() {
            let airlineDelays = {}; // Object to store airlines and their
delay counts

            jsonFlights.forEach(flight => {
                if (flight['schedueTime'] !== flight['actualTime']) {
                    const airline = flight['operatorLong'];
                    if (!airlineDelays[airline]) {
                        airlineDelays[airline] = 1;
                    } else {
                        airlineDelays[airline]++;
                    }
                }
            });

            let maxDelays = 0;
            let mostLateAirline = '';
            for (const airline in airlineDelays) {
                if (airlineDelays[airline] > maxDelays) {
                    maxDelays = airlineDelays[airline];
                    mostLateAirline = airline;
                }
            }

            // Update and show the late airline info section
            lateAirlineText.innerHTML = `The most late airline is
${mostLateAirline} with ${maxDelays} delays.`;
            showElement(lateAirlineInfo);
            // After setting the innerHTML of lateAirlineText and making
lateAirlineInfo visible
            lateAirlineInfo.classList.add('highlight-active');
            // Smoothly scroll to the lateAirlineInfo div
            lateAirlineInfo.scrollIntoView({ behavior: 'smooth' });
        }
```

קטע קוד זה הוא פונקציה בשם `findMostLateAirline` שמחשבת את חברת התעופה עם הכי הרבה עיכובים בהתבסס על נתוני JSON נתון של מידע טיסה.

כניסות

- `jsonFlights`: מערך של אובייקטי טיסה המכילים מידע על זמני טיסה מתוכננים ואמיתיים.

זרימה

1. אתחל אובייקט ריק בשם `airlineDelays` כדי לאחסן את חברות התעופה ואת ספירת העיכובים שלהן.
2. חזור על כל אובייקט טיסה במערך 'jsonFlights'.
3. בדקו אם הזמן המתוכנן (`schedueTime`) אינו שווה לזמן בפועל (`actualTime`) לטיסה.
4. אם יש עיכוב, קבלו את שם חברת התעופה (`operatorLong`) ובדקו אם הוא כבר קיים באובייקט `airlineDelays`.
5. אם חברת התעופה לא קיימת באובייקט `airlineDelays`, הוסף אותו עם ספירת עיכובים של 1. אחרת, הגדל את ספירת העיכובים עבור אותה חברת תעופה.
6. מצא את חברת התעופה עם ספירת העיכובים המקסימלית על ידי איטרציה דרך האובייקט `airlineDelays`.
7. עדכן את תוכן הטקסט של רכיב `lateAirlineText` עם חברת התעופה המאוחרת ביותר וספירת העיכובים שלה.
8. הצג את האלמנט 'lateAirlineInfo'.
9. הוסף את מחלקת ה-`highlight-active` `CSS` לאלמנט `lateAirlineInfo`.
10. גלול בצורה חלקה לאלמנט `lateAirlineInfo`.

פלטים

- הפונקציה מעדכנת את תוכן הטקסט של אלמנט `lateAirlineText` עם חברת התעופה המאוחרת ביותר וספירת ההשהיה שלה.
- האלמנט `lateAirlineInfo` מוצג ומודגש עם מחלקת ה-`highlight-active` `CSS`.
- העמוד גולל בצורה חלקה לאלמנט `lateAirlineInfo`.

8. אנימציית ענן: אנימציית רקע של עננים נעים (div.clouds) מוסיפה אלמנט ויזואלי הקשור לנושא טיסות ושדות תעופה. זה מושג באמצעות אנימציות CSS.

```
.clouds {
            position: fixed;
            top: 0;
            left: 0;
            width: 200%;
            height: 100%;
            z-index: -1;
            /* Ensure clouds are behind content */
            background-image: url('../Images/Cloud1.png'),
url('../Images/Cloud2.png');
            background-position: 0 50%, 100% 50%;
            background-size: 50% auto;
            animation: moveClouds 60s linear infinite;
            will-change: transform;
            /* Hint to the browser for optimization */
        }

        @keyframes moveClouds {
            0% {
                transform: translateX(0);
            }

            100% {
                transform: translateX(-100%);
            }
        }
```

קטע הקוד מגדיר מחלקת CSS הנקראת "clouds" שיוצרת אפקט רקע של עננים נעים. הוא משתמש בשתי תמונות ענן ומפעיל אותן כדי לנוע אופקית על פני המסך.

זרימה

1. מחלקת ה-CSS "clouds" מוגדרת עם המאפיינים הבאים:

- `position: fixed;` - מתקן את מיקום האלמנט על המסך
- `top: 0;` - ממקם את האלמנט בחלק העליון של המסך
- `left: 0;` - ממקם את האלמנט בצד שמאל של המסך
- `width: 200%;` - מגדיר את רוחב האלמנט לכפול מרוחב המסך
- `height: 100%;` - מגדיר את גובה האלמנט לגובה המלא של המסך
- `z-index: -1;` - מגדיר את סדר הערימה של האלמנט להיות מאחורי תוכן אחר

- `background-image: url('../Images/Cloud1.png'), url('../Images/Cloud2.png');` - מגדיר את תמונות הרקע של האלמנט לשתי תמונות ענן

- `background-position: 0 50%, 100% 50%;` - ממקם את תמונות הרקע במרכז אופקית ואנכית

- `background-size: 50% auto;` - מגדיר את גודל תמונות הרקע ל-50% מהרוחב והגובה האוטומטי של האלמנט

- `animation: moveClouds 60s linear infinite;` - מחיל את האנימציה "moveClouds" על האלמנט, עם משך של 60 שניות, פונקציית תזמון ליניארי וחזרה אינסופית

- `will-change: transform;` - רמז לדפדפן שהמאפיין "transform" ישתנה, מה שמאפשר אופטימיזציה

2. כלל `@keyframes moveClouds` מוגדר כדי לציין את התנהגות האנימציה:

- `0%` - מייצג את נקודת ההתחלה של האנימציה

- `transform: translateX(0);` - מגדיר את התרגום האופקי הראשוני של האלמנט ל-0

- `100%` - מייצג את נקודת הסיום של האנימציה

- `transform: translateX(-100%);` - מגדיר את התרגום האופקי הסופי של האלמנט ל-100% (מזיז את האלמנט לחלוטין מהמסך שמאלה)