לולאות

עד עכשיו למדנו לכתוב קוד בצורה יחסית פשוטה:
הקוד רץ בצורה כרונולוגית - 'מלמעלה למטה', שורה אחר שורה לאורך ריצת התוכנית, עד סוף קטע הקוד שלנו, לעיתים השתמשנו בתנאים על מנת לבצע רק חלק מסוים מהקוד.
הבעיה מתחילה כאשר אנחנו רוצים לחזור על קטע קוד מסוים כמות מסוימת של פעמים - באמצעות הידע שברשותנו עד כה, נאלץ לכתוב את אותו קטע הקוד מחדש ובכך לחזור על קוד!

למשל, אם אני ארצה להדפיס "Hello World!" 4 פעמים, אני אצטרך לכתוב את הפקודה console.log("Hello World!"); 4 פעמים.

```
console.log("Hello World!");
console.log("Hello World!");
console.log("Hello World!");
console.log("Hello World!");
```

אבל מה אם נרצה להדפיס הודעה מסוימת 1000 פעמים או מספר פעמים לפי **בחירת** משתמש? (דרך קלט מסוים).

בדיוק בשביל זה יש לנו אפשרות להשתמש בלולאות!

מהי לולאה?

לולאה – טכניקת תכנות המבצעת רוטינה/חזרה על קטע מספר רב של פעמים, עד שלב מסוים בו הלולאה עוצרת. אם ישנה פעולה שנרצה שתתבצע בצורה מחזורית מספר רב של פעמים, נשתמש בלולאה.
ישנם כמה סוגי לולאות ואנחנו נלמד על השניים הנפוצים:

1. לולאת תנאי
2. לולאת מונה בקרה

לולאת תנאי

תפקיד פקודת לולאת תנאי - פקודת לולאת התנאי מאפשרת ביצוע פקודה או רצף פקודות באופן מחזורי.

תנאי הלולאה - נכתב בתוך הסוגריים, הפקודות ימשיכו לחזור על עצמן, כל עוד התנאי בתוך הסוגריים נכון (ניתן להשתמש גם במשתנה בוליאני).

מבנה לולאות while

```
while (תנאי) {

    //רצף פקודות

}
```

מספר דגשים נוספים:

- כאשר התנאי לא מתקיים, לא ניכנס ללולאה, והתוכנית תמשיך לפקודה הראשונה שאחרי הלולאה.
- חשוב לא לשכוח לתחום את כל פקודות הלולאה בין סוגריים מסולסלים {}.
- חשוב לא לשכוח לתחום את התנאי עצמו בסוגריים רגילים ().

לדוגמה :

```
let nNum = 0;
while (nNum < 4) {
    console.log("Hello World!");
    nNum++;
}
```

בקוד זה המחרוזת "Hello World!" תודפס 4 פעמים, למרות שפקודת ההדפסה כתובה פעם אחת בלבד!
זאת מפני שהקוד אינו רץ מלמעלה למטה לפי הסדר, אלא בכל פעם הוא בודק האם התנאי בלולאת ה while-עדיין מתקיים - ואז חוזר על קטע הקוד שבתוך הלולאה מחדש.

- **while = "כל עוד"** <-

כלומר, כל עוד התנאי מתקיים (nNum < 4), קטע הקוד בתוך הלולאה יחזור על עצמו.

לולאת - while מונה

אחד מסוגי לולאות הwhile הוא לולאה המבוססת על משתנה מונה מסוים. מגדירים לו ערך התחלתי מחוץ ללולאה, ומשנים אותו בתוך הלולאה. תנאי הלולאה יהיה תלוי במונה כך שהלולאה תתרחש כל עוד המונה קטן מ/ גדול מ/ שווה לערך מסוים.

*בלולאה זו חייבים לקדם את המונה/ המשתנה הקשור לתנאי בתוך הלולאה, או לדאוג לשינוי התנאי כדי למנוע מצב של לולאה אינסופית.

<u>לדוגמה:</u>

שימוש בלולאת while על מנת להדפיס מספרים שלמים הקטנים מ-10:

```
let nLoopCounter = 1;
while (nLoopCounter <= 10) {
    console.log(nLoopCounter);
    nLoopCounter++;
}
```

1. נגדיר ערך מסוים בו נשתמש לקבוע את המצב בו נכנס/לא נכנס ללולאה. נגדיר משתנה nLoopCounter שיאותחל בערך 1.
2. לאחר מכן, נגיע ללולאת while ובתוך הסוגריים יוגדר 'תנאי העצירה' של הלולאה.

- **אם התנאי מתקיים** (אם nLoopCounter קטן או שווה ל10) - נכנס לתוך הבלוק, ההדפסה תבוצע והמשתנה יגדל ב1.
- **אם התנאי אינו מתקיים** (אם nLoopCounter גדול מ10) – הקוד שכתוב בתוך הבלוק של הלולאה (סוגריים מסולסלים {}) לא יתבצע, והריצה ממשיכה כרגיל בסדר כרונולוגי.

שימו ♥!

הפקודה ++nLoopCounter הכרחית לביצוע הלולאה – פקודה זו מקדמת את המשתנה count ב-1. כך, לאחר 10 ריצות חוזרות של הלולאה, תנאי הלולאה לא יתקיים יותר והלולאה תסתיים.

כל עוד נתכנן את הקוד שלנו ונדאג להשתמש בלולאות בצורה נכונה, קטע הקוד בלולאה יסתיים בשלב מסוים – נרצה להימנע ממצב שבו התנאי תמיד מתקיים כי נגיע למצב של **לולאה אינסופית** (לולאה שתמיד תתבצע). במצב זה, הריצה של הקוד לא תמשיך ורק תבצע את קטע הקוד שבתוך הלולאה פעם אחר פעם.

פירוט ריצת הקוד של לולאת ה while בדוגמה

- המשתנה nLoopCounter מאותחל בערך 1.
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא כן ולכן נכנסים ללולאה.
- מודפסת ההודעה: Hello World **פעם אחת** ומקדמים את המשתנה nLoopCounter,
  מכאן שערכו: nLoopCounter = 1,
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא כן ולכן נכנסים ללולאה.
- מודפסת ההודעה: Hello World **פעם שניה** ומקדמים את המשתנה nLoopCounter,
  מכאן שערכו: nLoopCounter = 2.
- כך הלולאה תמשיך עד שתגיע ל-nLoopCounter = 11.
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא **לא** ולכן **יוצאים** מהלולאה.
- ממשיכים כרגיל בהמשך ריצת התוכנית עד סופה.

***בכל פעם שנרצה לעקוב אחרי ריצת לולאה, נעשה זאת בדרך המוצגת למעלה או דרך טבלת מעקב.

פירוט ריצת הקוד של לולאת ה while דרך טבלת מעקב

מתחילים מ nLoopCounter = 1.

שלב	nLoopCounter <= 10	פלט	nLoopCounter++
מעבר ראשון	True	1	2
מעבר שני	True	2	3
מעבר שלישי	True	3	4
מעבר רביעי	True	4	5
מעבר חמישי	True	5	6
מעבר שישי	True	6	7
מעבר שביעי	True	7	8
מעבר שמיני	True	8	9
מעבר תשיעי	True	9	10
מעבר עשירי	True	10	11
מעבר 11	False	*אין כניסה ללולאה*	*אין כניסה ללולאה*

תרגול

בנו לולאה המסכמת את המספרים שבין 50 ל 100 כולל, ומדפיסה את הסכום.

לולאת מונה בקרה (for)

פקודת לולאת מונה בקרה הנה בפועל לולאת תנאי משוכללת יותר.
לולאת מונה בקרה מאפשרת ביצוע פקודה או רצף פקודות באופן מחזורי מספר קבוע **<u>וידוע מראש</u>** של פעמים.

<u>מבנה לולאת for:</u>

```
for ( מונה = ערך התחלתי; מונה < ערך סופי; קידום המונה) {

        //תוכן הלולאה

}
```

<u>היא מכילה 3 חלקים:</u>

1. הצהרה ונתינת ערך התחלתי של המונה: counter = value.
2. תנאי יציאה מהלולאה (ערך מקסימלי): counter < maxValue.
   חייבים שתנאי הלולאה יתייחס למונה.
3. צעדי המונה: counter = counter + 1.
   נכתוב בקיצור counter++.

שימו לב שהפרמטרים מופרדים ע"י נקודה-פסיק.

- **for = "עבור כל"** ->
  עבור כל מונה שקטן מהערך הסופי = הלולאה תתקיים.

<u>למה נצטרך לולאות for?</u>

נבצע את אותה הדוגמה שביצענו קודם באמצעות שימוש בלולאת תנאי (הדפסת מספרים הקטנים מ -10) כדי לראות את היתרון של לולאת מונה בפעולות מסוג זה.

```
for (let nLoopCounter = 1; nLoopCounter <= 10; nLoopCounter++) {
    console.log(nLoopCounter);
}
```

ניתן לראות שלולאה זו יעילה יותר, גם הגדרת המשתנה וגם קידום המשתנה נמצאים כחלק מכותרת מהלולאה.

***כאשר מספר הפעמים שנרצה לבצע את הלולאה ידוע מראש, <u>תמיד</u> נרצה להשתמש בלולאת for כדי לכתוב את הלולאה ולא בלולאת while מונה.**

פירוט ריצת הקוד של לולאת ה for בדוגמה:

- המשתנה nLoopCounter מאותחל בערך 1.
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא כן ולכן נכנסים ללולאה.
- מודפסת ההודעה: Hello World **פעם אחת.**
- מקדמים את המשתנה nLoopCounter, מכאן שערכו: nLoopCounter = 1.
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא כן ולכן ממשיכים ונכנסים פעם נוספת ללולאה.
- מודפסת ההודעה: Hello World **פעם שניה.**
- מקדמים את המשתנה nLoopCounter, מכאן שערכו: nLoopCounter = 2.
- כך הלולאה תמשיך עד שתגיע ל- nLoopCounter = 11.
- מתבצעת בדיקה האם nLoopCounter <= 10, התשובה היא **לא** ולכן **יוצאים** מהלולאה.
- ממשיכים כרגיל בהמשך ריצת התוכנית עד סופה.

***בכל פעם שנרצה לעקוב אחרי ריצת לולאה, נעשה זאת בדרך המוצגת למעלה או דרך טבלת מעקב.

דגש חשוב !

למרות שכל הפעולות נכתבות בכותרת הלולאה, את קידום המשתנה, נעשה רק בסיום הפקודות שרשומות בתוך הבלוק של הלולאה {}. כלומר, הסדר הוא- בדיקת תנאי הלולאה, לאחר מכן הפקודות הרשומות בתוכה ורק אז קידום המונה ובדיקת התנאי פעם נוספת.

ניתן גם להגדיר את הלולאה בצורה יורדת:
ערך התחלתי מקסימלי, שבכל צעד יורד עד שמגיע לערך המינימלי:

```
for (let nCounter = 10; nCounter > 0; nCounter--)  {
    console.log(nCounter);
}
```

השוואה בין for ל while

כל לולאת for ניתן לפתור גם באמצעות לולאת while.
שימוש בלולאת for יכול להיות פשוט ונוח יותר במצבים מסוימים.

לולאות for היא סוג של לולאת while למעשה מדובר בלולאת while עם מונה שמתקדם בצורה אוטומטית.
כאשר מספר הפעמים לריצת הלולאה **ידוע** מראש, נעדיף להשתמש בלולאת **for**
כאשר מספר הפעמים לריצת הלולאה **אינו ידוע** מראש, או אינו תלוי במונה מסוים (למשל תלוי במשתנה בוליאני בלבד), נעדיף להשתמש בלולאת while

תחום	**לולאת תנאי**	**לולאת מונה**
קידום המונה	בגוף הלולאה	בפקודת ה for (בכותרת הלולאה)
אפשרות לביצוע מספר לא ידוע של כניסות ללולאה	אפשרי	לא אפשרי

סטנדרטים

- בלוקים של קוד יתחמו בסוגריים מסולסלים {}.
- פתיחת סוגריים מסולסלים תהיה באותה השורה של כותרת הלולאה.
- יהיה קיים ריווח בין ה for/while והסוגריים המסולסלים.
- יהיה קיים ריווח בין ה for/while ופתיחת הסוגריים ().

סיכום

לולאות הן מרכיב הכרחי וחשוב בכתיבת הקוד שלנו, הן מאפשרות לנו לכתוב תכניות מורכבות יותר, לחסוך בקוד/עבודה מיותרת, ומאפשרות יכולות שלא היו אפשריות לפני ללא חזרה משמעותית על קוד.