מערך חד- ממדי

כאשר אנחנו יוצרים משתנה לא מאותחל מסוג המחלקה Array<elementType> , לדוגמא:

```
let x: Array<number>;
```

x הוא משתנה מסוג reference type שמוגדר בזיכרון ויכול להכיל הפניה לאובייקט שהוא מערך.

אבל עדיין לא יצרנו את האובייקט עצמו (לא היה שימוש בnew), ולכן כרגע x מכיל undefined, ולא מכיל הפניה לשום אובייקט מערך.

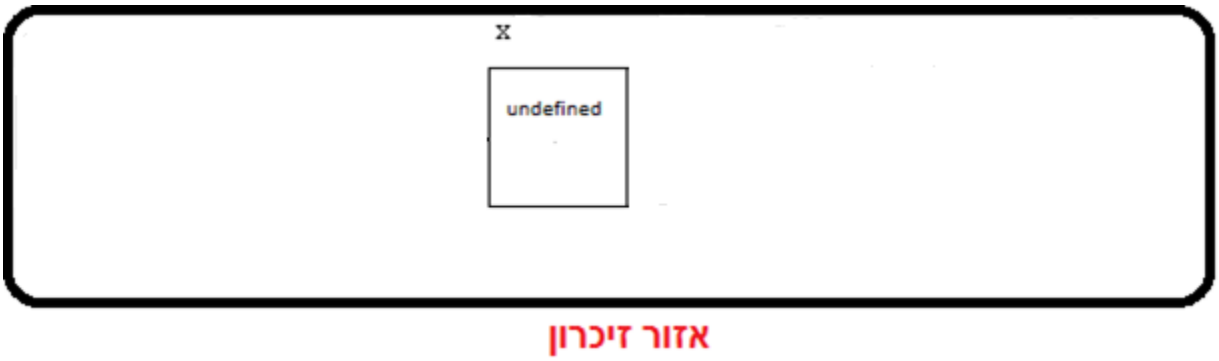
כעת נוכל להבין מדוע כשנריץ את הקוד:

```
let x: Array<number>;
document.write("x.length = "+x.length);
```

נקבל את שגיאת זמן הריצה הבאה:

```
Uncaught TypeError: Cannot read property 'length' of undefined
    at app.js:2
>
```

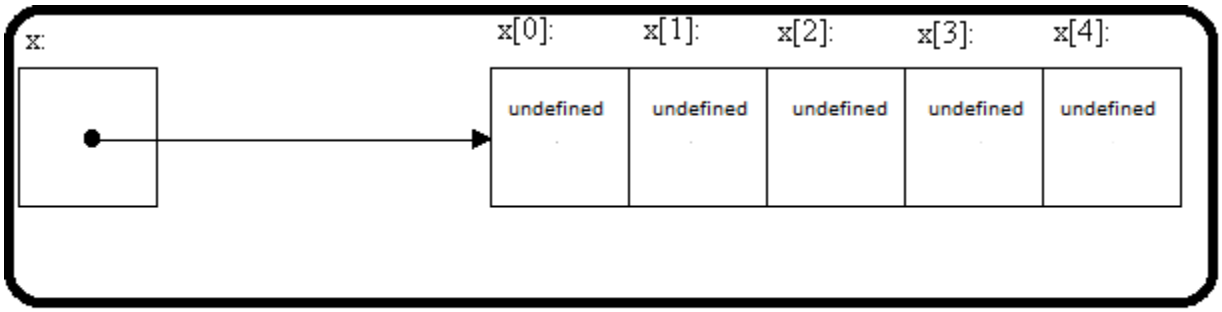
הסיבה היא מכיוון שהזיכרון כרגע נמצא במצב הבא:



אך, כאשר ניצור מערך ע"י המילה new:

```
let x: Array<number> = new Array<number>(5);
```

יצרנו בזיכרון מערך בגודל חמישה תאים, והמשתנה x יכיל בתוכו הפניה לאותו אובייקט מערך שנוצר, הזיכרון יראה כך:



אזור זיכרון

ננסה להריץ עכשיו את הקוד הבא:

```
let x: Array<number> = new Array<number>(5);
document.write("x.length = "+x.length);
```

ונקבל תוצאה תקינה.
הפלט יהיה:

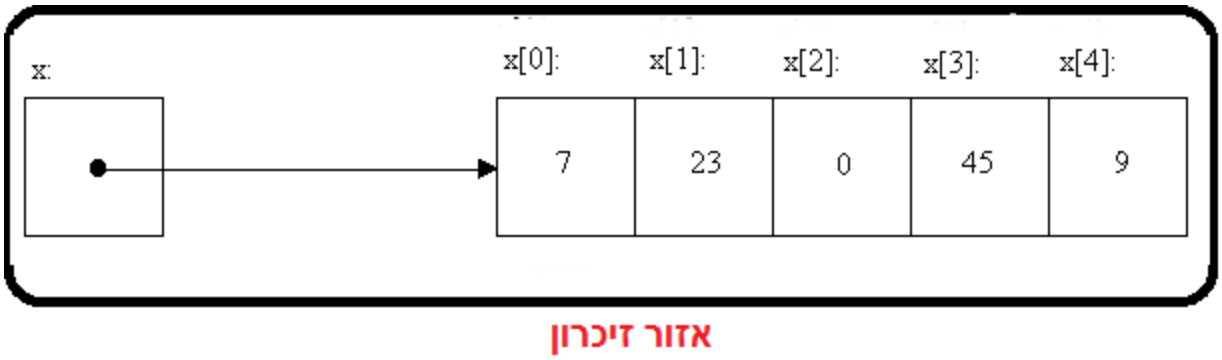
x.length = 5

למעשה, בהגדרת המערך כ- `Array<number>`, הגדרנו שכל תא מתאי המערך הוא כמו משתנה מספרי, יכול להכיל ערך מספרי (number).
אבל, בשלב זה, שרק יצרנו את המערך, כל תא במערך יכיל undefined (הערך הדיפולטיבי של משתנה שנוצר ולא מאותחל), מכיוון שעדיין לא הצבנו לכל תא תוכן.

נבצע הצבה של ערכים מספריים לתאי המערך:

```
let x: Array<number> = new Array<number>(5);
x[0] =7;
x[1] =23;
x[2] =0;
x[3] =45;
x[4] =9;
```

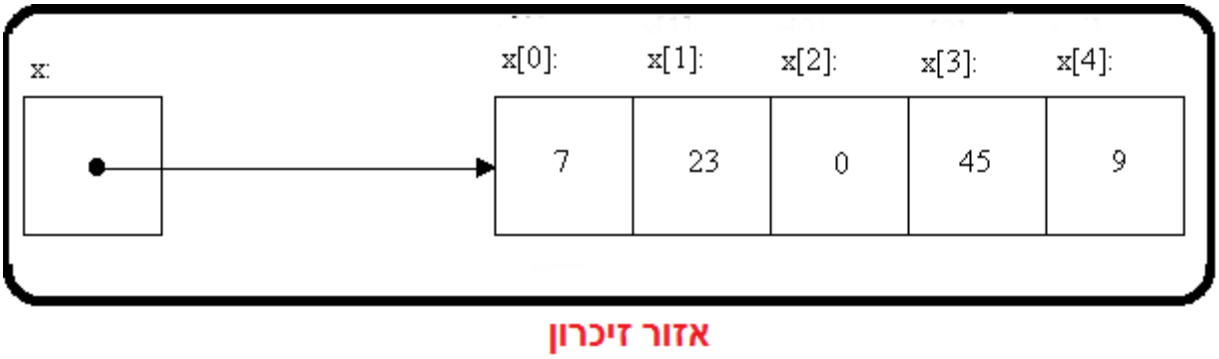
ונקבל את התוצאה הבאה בזיכרון:



כמובן, שנוכל גם להשתמש בתחביר הבא:

```
let x: Array<number> =[7,23,0,45,9];
```

שמאחורי הקלעים יצור לנו אובייקט חדש של מערך בן חמישה תאים, בו כל תא הוא מסוג מספרי. ולתוך כל תא יוכנס ערך בהתאם לערכים שהזנו בשורת ההגדרה. ולכן, התוצאה תהיה אותה מפת זיכרון:



<u>לסיכום:</u> ביצירת מערך כאשר נגדיר את סוג המשתנה של תאי המערך, ייווצרו במערך תאים **המסוגלים** להכיל ערך מטיפוס המשתנה שהגדרנו, ויכילו את הערך undefined עד שיושם לתוכם ערך. (ההשמה יכולה להתבצע בשורת ההגדרה של המערך, או לאחר יצירת המערך ע"י הצבה לתא מסוים).

מערך דו – ממדי

ב-typescript כל מערך דו ממדי, מכיל:

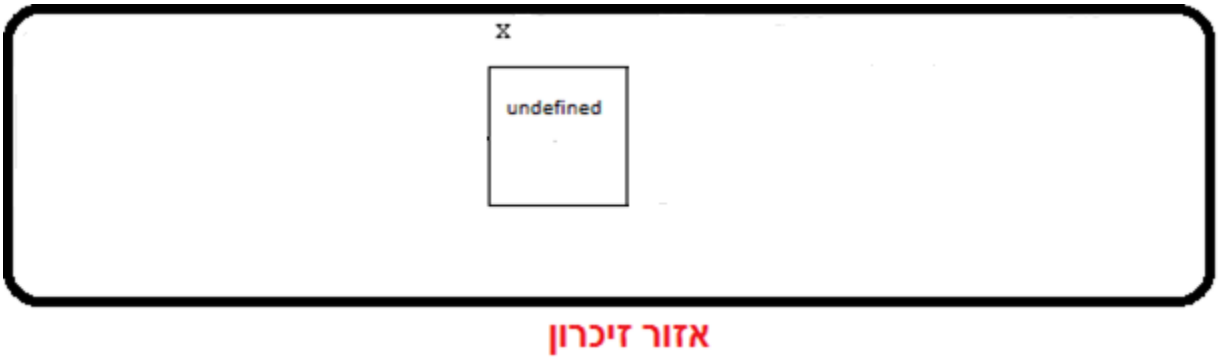
1. **מערך ראשי-** מערך חד ממדי בו על תא מכיל הפניה למערך משנה
2. **מערכי משנה-** מערך חד ממדי בו כל תא מכיל ערך בודד של הטיפוס המוגדר למטריצה

שלבי יצירת מערך דו- ממדי:

שלב ראשון – הגדרת משתנה מטיפוס מטריצה מספרית

```
let x: Array<Array<number>>;
```

בשלב זה המשתנה x לא מכיל הפניה לשום מטריצה, ומאותחל לערך הדיפולטיבי undefined, ולכן הזיכרון נמצא במצב הבא:

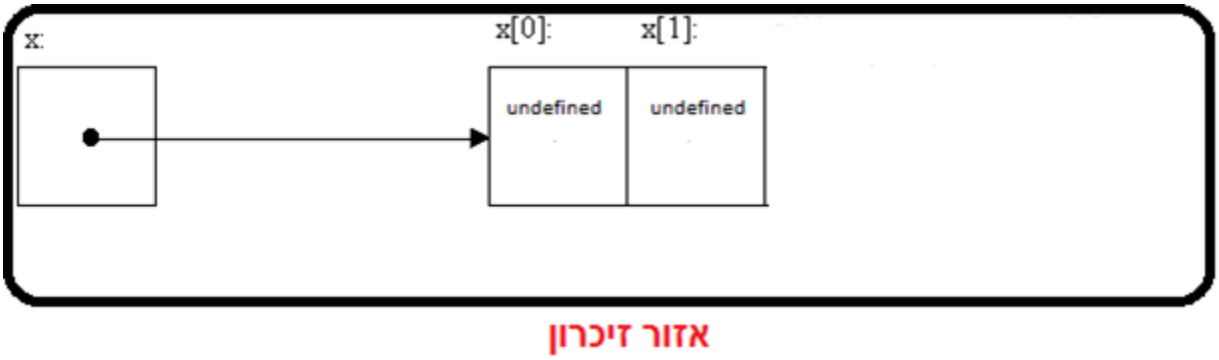


שלב שני – יצירת אובייקט של מערך חד ממדי (בו כל תא מצביע למערך משנה)

```
x = new Array<Array<number>>(2);
```

בשלב זה המשתנה x מכיל הפניה למערך בעל שני תאים.

כל תא במערך עליו x מצביע, יכול להכיל הפניה למערך חד ממדי מסוג מספרי, אך בשלב זה עדיין לא ביצענו לתאים השמה של הפניה למערך חד ממדי מספרי, ולכן הם מאותחלים לערך הדיפולטיבי undefined, והזיכרון נמצא במצב הבא:

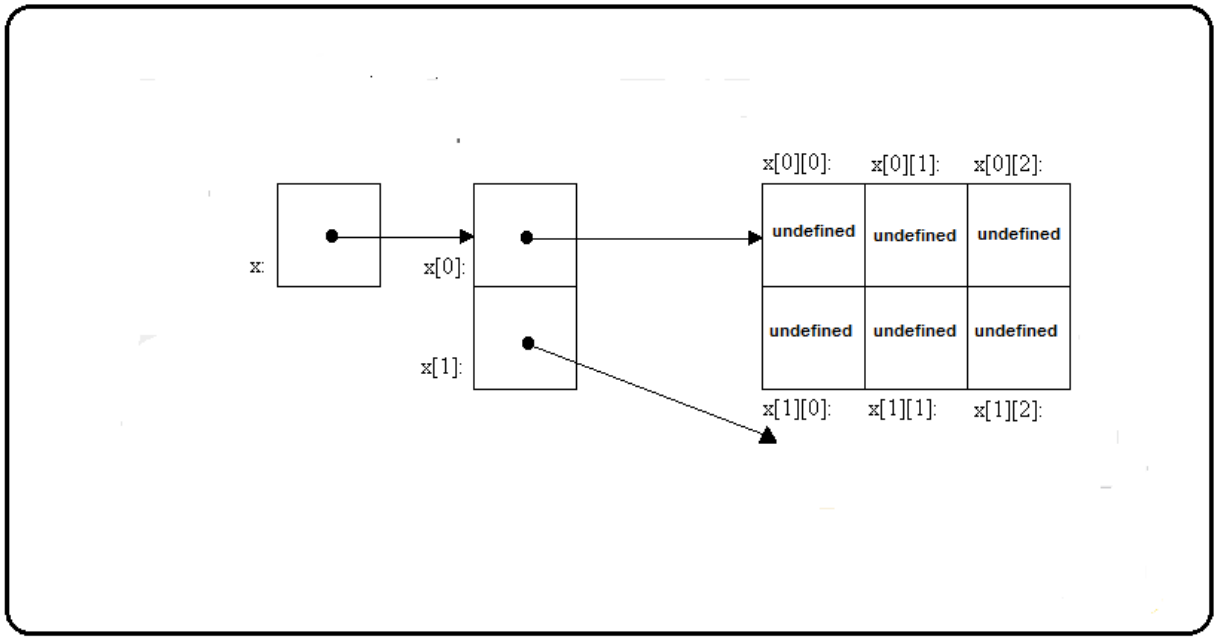


שלב שלישי – יצירת אובייקט של מערך חד ממדי (מסוג מספרי) עבור כל תא במערך הראשי

```
x[0] = new Array<number>(3);
x[1] = new Array<number>(3);
```

בשלב זה המשתנה x מכיל הפניה למערך בעל שני תאים.

כל תא במערך עליו x מצביע, מכיל הפניה למערך חד ממדי מסוג מספרי בגודל שלושה תאים, אך בשלב זה עדיין לא ביצענו לתאים במערכים המשניים השמה של ערך מספרי, ולכן הם מאותחלים לערך הדיפולטיבי undefined, והזיכרון נמצא במצב הבא:



אזור זיכרון

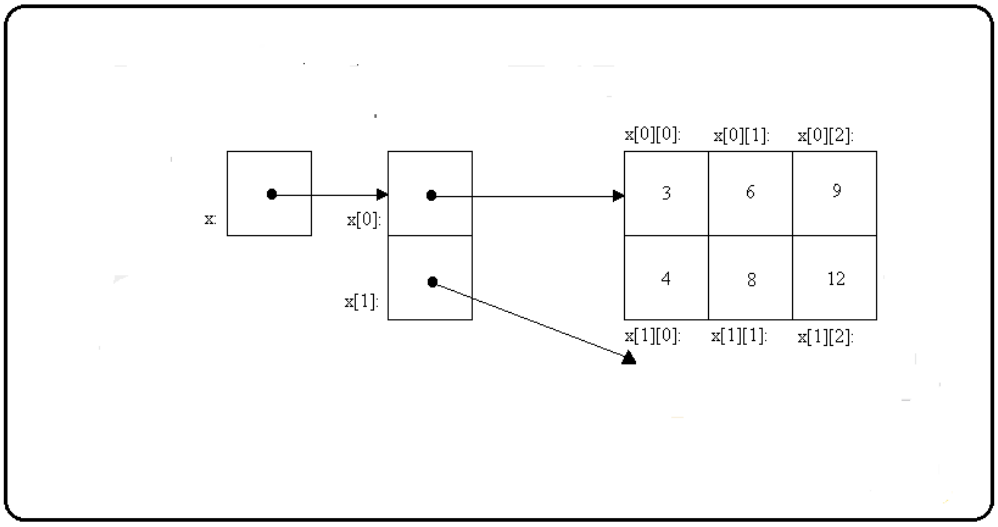
שלב רביעי – השמת ערך מספרי לכל תא במטריצה

```
x[0][0] = 3;
x[0][1] = 6;
x[0][2] = 9;
x[1][0] = 4;
x[1][1] = 8;
x[1][2] = 12;
```

בשלב זה המשתנה x מכיל הפניה למערך בעל שני תאים.

כל תא במערך עליו x מצביע, מכיל הפניה למערך חד ממדי מסוג מספרי בגודל שלושה תאים, וכל תא מהמערך החד ממדי משני מכיל את הערך המספרי שהצבנו לתוכו.

הזיכרון נמצא במצב הבא:



אזור זיכרון

לסיכום: להלן התכנית המלאה כולל לולאה מקוננת להדפסת תוכן המטריצה

```
let x = new Array<Array<number>>(2);
x[0] = new Array<number>(3);
x[1] = new Array<number>(3);

x[0][0] = 3;
x[0][1] = 6;
x[0][2] = 9;
x[1][0] = 2;
x[1][1] = 4;
x[1][2] = 8;

document.write(`<br/>Martix x with ${x.length} rows: `);
for (let i: number = 0; i < x.length; i++) {
    document.write(`<br/><br/> row number ${i} with ${x[i].length} elements: `);
    for (let j: number = 0; j < x[i].length; j++) {
        document.write(x[i][j] + " ");
    }
}
```