העברת ערכים לפונקציה ע"י Value Type and Reference type

1) **העברת פרמטר by value**- הקוד שקורא לפונקציה ושולח לה **ערך של תוכן משתנה** שתוכנו **מועתק** לתוך הפרמטר הלוקלי של הפונקציה

העברת פרמטר by value-תתבצע כאשר נעביר משתנים מסוג number, boolean

לדוגמה

בקוד הבא יצרנו משתנה מסוג number, העברנו אותו לפונקציה שמקדמת את תוכנו, והדפסנו את המשתנה לפני הקריאה לפונקציה, ואחרי הקריאה לפונקציה, כדי לראות את השפעת הפונקציה על המשתנה.

```
//////////////MAIN SECTION///////////////
let numOrginal: number = 9;

document.write("numOrginal before function: " + numOrginal + "<br/>");
incNum(numOrginal);
document.write("numOrginal after function: " + numOrginal + "<br/>");


//////////////FUNCTION SECTION///////////////
function incNum(numParam: number): void {
    numParam++;
}
```

הפלט שנקבל כאשר נריץ את התוכנית יהיה:

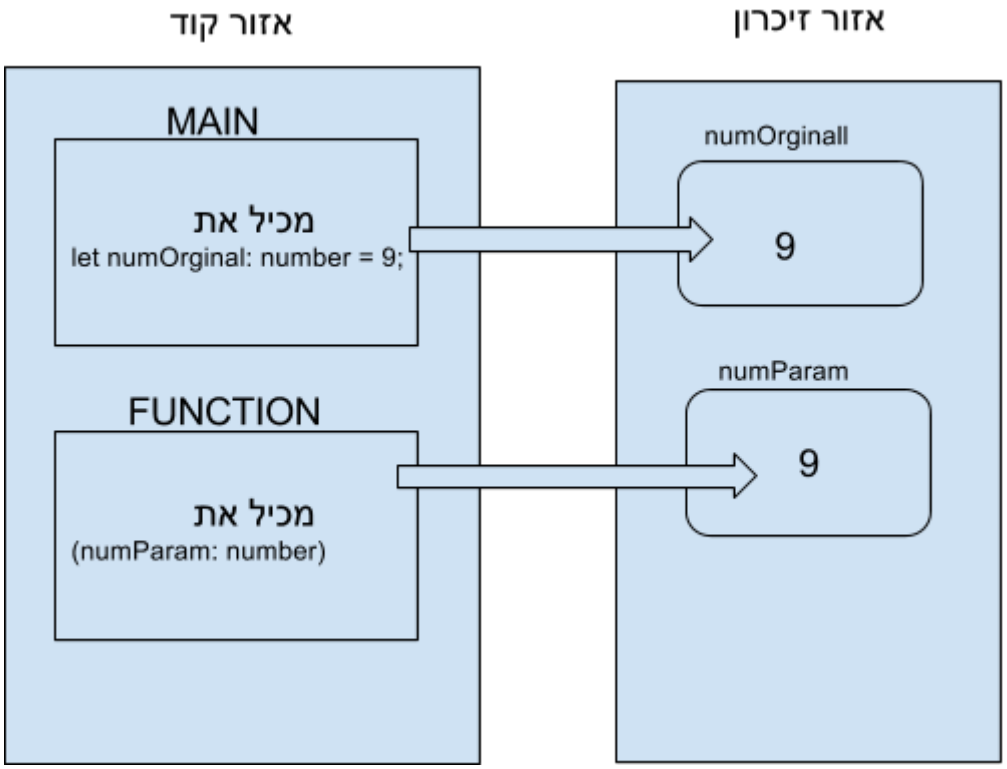
numOrginal before function: 9
numOrginal after function: 9

כלומר אנו יכולים לראות שהשינוי שביצענו בתוך הפונקציה על המשתנה, לא נשמר על המשתנה המקורי אתו ביצענו את הקריאה לפונקציה.

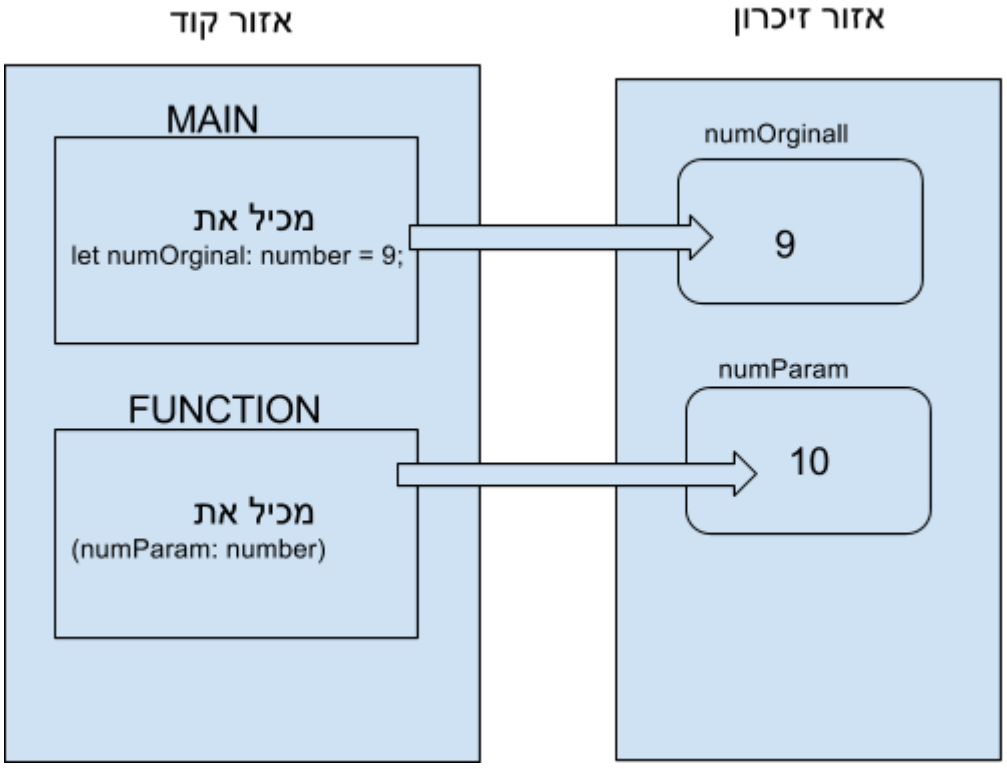
התופעה הזו התרחשה מכיוון שהפרמטר שהועבר לפונקציה היה בעצם העתקה של תוכן המשתנה numOrginal , כלומר לתוך הפרמטר של הפונקציה הועבר הערך 9 שהיה בתוך המשתנה, הערך הזה נכנס לתוך המשתנה הפרטי של הפונקציה שנקרא numParam, ועליו בוצעה הפעולה של הגדלת הערך, כך שהמשתנה numOrginal לא הושפע בכלל.

ניצור מעין מפת זיכרון של התוכנית כדי להמחיש את הרעיון:

בעת הקריאה לפונקציה incNum:



כלומר אנו רואים שלחלק הMAIN יש משתנה משלו, ולפונקציה יש משתנה משלה
בעת הקריאה לפונקציה מועתק הערך מתוך המשתנה numOrginal אל numParam
ולכן כשאשר הפונקציה תקדם את תוכן המשתנה נקבל את המפה הבאה:



וזו הסיבה שכאשר נדפיס את מתוך הMAIN לא נראה בו שום הבדל, כיוון שהשינוי נשמר רק על המשתנה הלוקלי של הפונקציה שנקרא numParam, ולא על המשתנה numOrginal עליו הMAIN מצביע.

2) **העברת פרמטר by reference**- הקוד שקורא לפונקציה ושולח לה ערך של **כתובת למשתנה מסוים בזיכרון**, הכתובת **מועתקת** לתוך הפרמטר הלוקלי של הפונקציה

העברת פרמטר by reference - תתבצע כאשר נעביר לפונקציה משתנה שהוא מערך

<u>לדוגמה</u>

בקוד הבא יצרנו משתנה מסוג number, העברנו אותו לפונקציה שמקדמת את תוכנו, והדפסנו את המשתנה לפני הקריאה לפונקציה, ואחרי הקריאה לפונקציה, כדי לראות את השפעת הפונקציה על המשתנה.

```
//////////////MAIN SECTION////////////////
let numArr: number[]=[1,2,3];

document.write("numArr before function: ");
printAllArray(numArr);

incNumArray(numArr);

document.write("numArr after function: ");
printAllArray(numArr);



//////////////FUNCTION SECTION////////////////
function printAllArray(arrayParam: Array<number>): void {
    document.write(arrayParam.toString() + "<br/>");

}

function incNumArray(numArrayParam: number[]): void {
    for (let i: number = 0; i < numArrayParam.length; i++) {
        numArrayParam[i]++;
    }
}
```

הפלט שנקבל כאשר נריץ את התוכנית יהיה:

numArr before function: 1,2,3

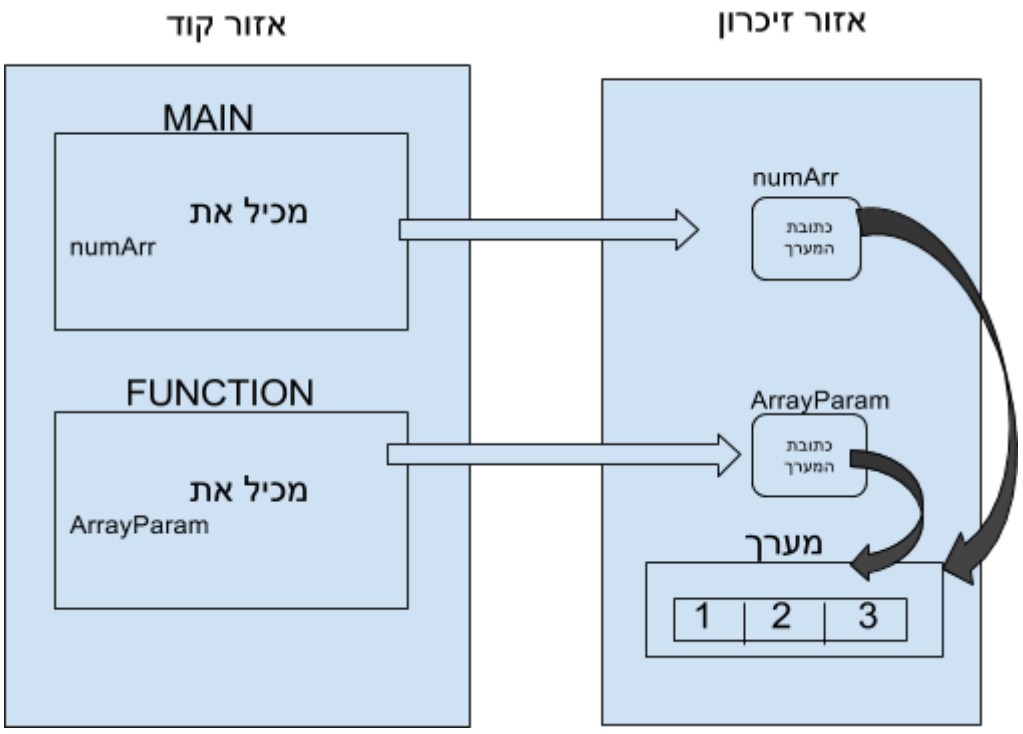
numArr after function: 2,3,4

כלומר אנו יכולים לראות שהשינוי שביצענו בתוך הפונקציה על המערך, נשמר על המערך המקורי אתו ביצענו את הקריאה לפונקציה.

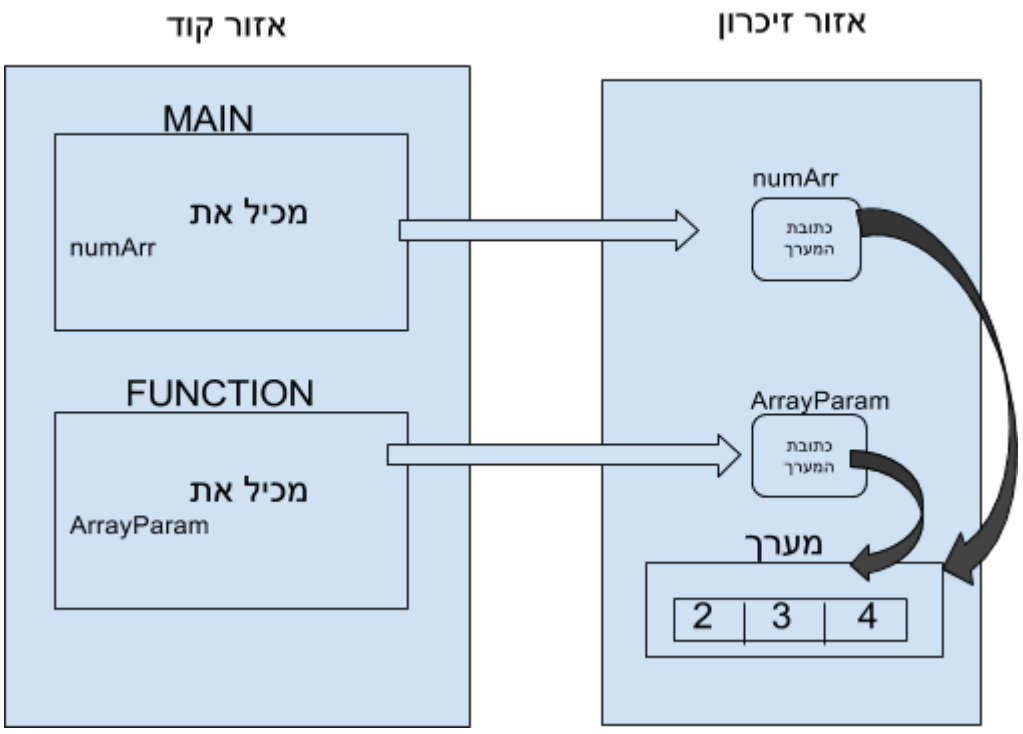
התופעה הזו התרחשה מכיוון שהפרמטר שהועבר לפונקציה היה בעצם העתקה של כתובת המערך, כלומר לתוך הפרמטר של הפונקציה הועבר הערך של הכתובת ההתחלתית של המערך בזיכרון.
הכתובת הזו הייתה שמורה בתוך המשתנה numArr, והועתקה לתוך המשתנה הפרטי של הפונקציה שנקרא numParam, והשינוי בוצע על אברי המערך, כך שהמערך הוא זה שגם הMAIN וגם הFUNCTION פונים אליו.

ניצור מעין מפת זיכרון של התוכנית כדי להמחיש את הרעיון:

בעת הקריאה לפונקציה incNumArray:



כלומר אנו רואים שלחלק הMAIN יש משתנה משלו, ולפונקציה יש משתנה משלה אבל שניהם מצביעים לאותו מקום בזיכרון שבו ממוקם המערך.
ולכן כשאשר הפונקציה תקדם את תוכן המשתנה נקבל את המפה הבאה:



וזו הסיבה שכאשר נדפיס את המערך מתוך הMAIN נראה את ההבדל.