תנאים בtypeScript

נושאי השיעור

1) המרת מחרוזת למספר
2) קלט לתוך מספר
3) אופרטורי קיצור עבור פעולות חשבוניות
4) משתנה const
5) הטיפוס Boolean
6) תנאי בסיסי
7) תנאי מורכב

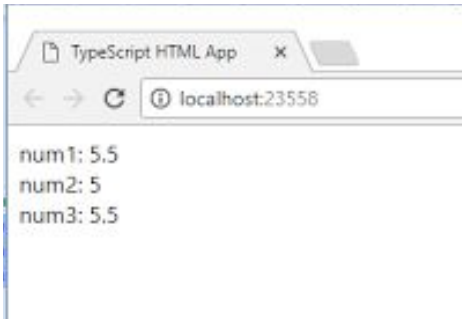
העברת ערך מחרוזת למספר

לא ניתן באמצעות הפונקציה prompt לקבל ערך שאינו מחרוזת, אבל בכדי שנוכל לבצע פעולות חשבון על משתנה, הוא חייב להיות מטיפוס מספרי, ולכן יש להעביר את הערך הנמצא בתוך משתנה מחרוזתי לתוך משתנה אחר מטיפוס מספרי.

נכיר 3 דרכים לבצע את ההמרה הזו:

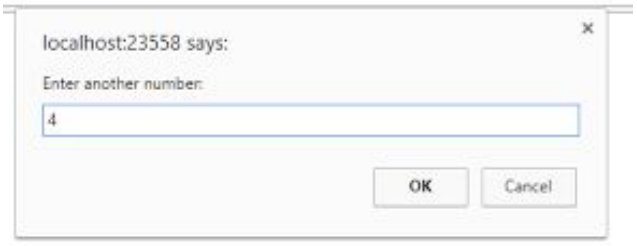
- parseInt - עבור מספרים שלמים
- parseFloat - עבור מספרים עשרוניים
- Number - עבור מספרים עשרוניים

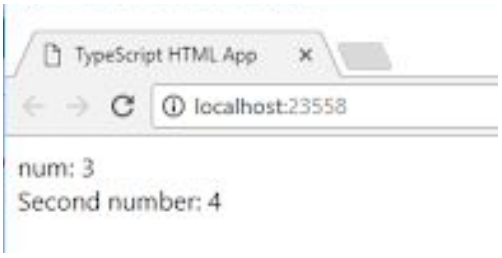
```
//05 - Convert string to number
let num1: number;
let num2: number;
let num3: number;
let str;

str = "5.5";

num1 = parseFloat(str); //num1=5.5
num2 = parseInt(str);   //num1=5
num3 = Number(str);          //num1=5.5

document.write("num1: " + num1 + "<br>");
document.write("num2: " + num2 + "<br>");
document.write("num3: " + num3 + "<br>");
```



קליטת ערך לתוך משתנה מספרי

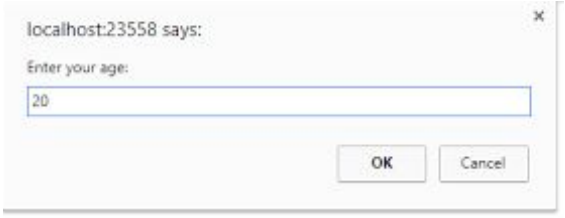
```
//06 - get number from user

let str: string;
let num: number;
str = prompt("Enter a number: ");
num = Number(str);
document.write("num: " + num + "<br>");


let num2: number;
num2 = Number(prompt("Enter another number: "));
document.write("Second number: " + num2);
```

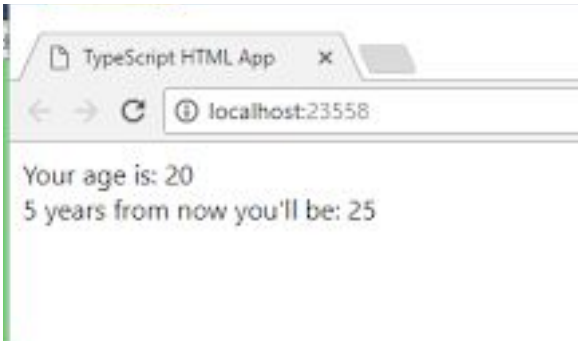




תרגיל: צרו משתנה וקלטו לתוכו את גיל המשתמש, הציגו לו את גילו הנוכחי ואת גילו בעוד 5 שנים

```
//07 - Get user age

let age: number;

age = Number (prompt("Enter your age: "));

document.write("Your age is: " + age + "<br>");
document.write("5 years from now you'll be: " + (age + 5));
```





אופרטורים מתמטיים על משתנים

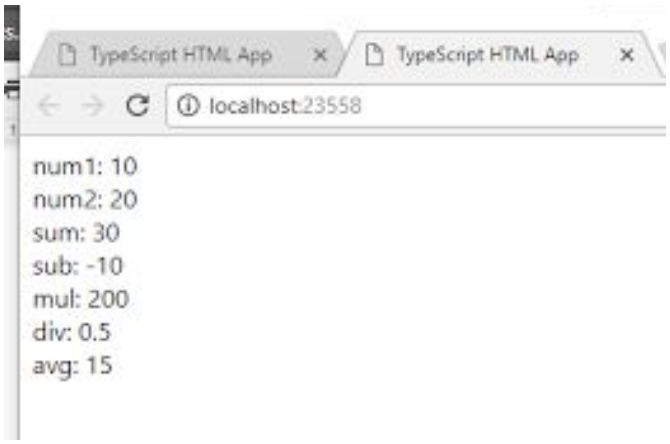
Now we can see that we can initialize a variable with a mathematical expression, when we create the variable:

אופרטור פעולה שאפשריים על מספרים נומריים:

- Addition **+**
- Subtraction **-**
- Multiplication *****
- Division **/**
- Modulus %

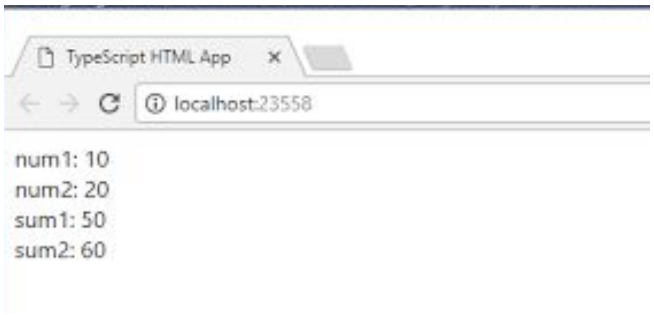
```
//02 - Math Operations

let num1: number;
let num2: number;

num1 = 10;
```

```
num2 = 20;

let sum: number = num1 + num2;
let sub: number = num1 - num2;
let mul: number = num1 * num2;
let div: number = num1 / num2;
let avg: number = sum / 2;

document.write("num1: " + num1 + "<br>");
document.write("num2: " + num2 + "<br>");
document.write("sum: " + sum + "<br>");
document.write("sub: " + sub + "<br>");
document.write("mul: " + mul + "<br>");
document.write("div: " + div + "<br>");
document.write("avg: " + avg + "<br>");
```



כמו בחשבון יש קדימות לאופרטורים *,/,% על פני האופרטורים +, -
ניתן לשנות את קדימות האופרטורים באמצעות סוגריים

```
//03 – Advanced Math Operations

let num1: number;
let num2: number;

num1 = 10;
num2 = 20;

let sum1: number = num1 + num2*2;
let sum2: number = (num1 + num2)*2;

document.write("num1: " + num1 + "<br>");
document.write("num2: " + num2 + "<br>");
document.write("sum1: " + sum1 + "<br>");
document.write("sum2: " + sum2 + "<br>");
```




```
//04 - Display Address

let country: string = "Israel";
let city: string = "Tel Aviv";
let street: string = "Hertzel";
let houseNumber: number = 17;
let zipCode: number = 12546;

document.write("Country: " + country + ", City: " + city + ", Street: " + street + ", House Number: " + houseNumber +
", Zip Code: " + zipCode);
```

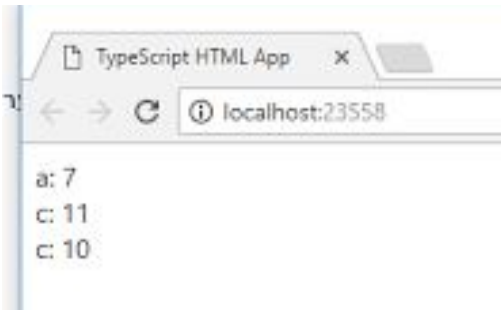
אופרטורי קיצור

אופרטורים לקיצור השמה משמשים כאשר נרצה להכניס ערך לתוך משתנה, והערך החדש הינו שינוי של הערך הקיים כבר במשתנה.

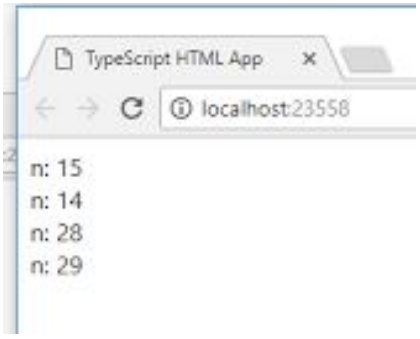
האופרטורים בהם נשתמש:

```
//15 - Short operators

let a: number, b: number;

a = 3;
b = 4;

a += b;

document.write("a: " + a + "<br>");

let c: number = 10;
```

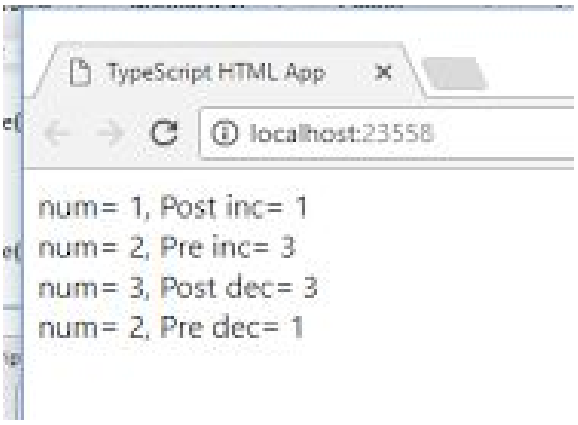
```
c++;
document.write("c: " + c + "<br>");
c--;
document.write("c: " + c);
```




```
//16 - Short operators exercise

let n: number = 10;

n += 5;

document.write("n: " + n + "<br>");

n--;

document.write("n: " + n + "<br>");

n *= 2;

document.write("n: " + n + "<br>");

n++

document.write("n: " + n + "<br>");
```




```
//17 - pre and pro
let num: number = 1;

document.write(`num= ${num}, Post inc= ${num++} <br/>`);
document.write(`num= ${num}, Pre inc= ${++num} <br/>`);
document.write(`num= ${num}, Post dec= ${num--} <br/>`);
document.write(`num= ${num}, Pre dec= ${--num} <br/>`);
```



קבועים

ניתן להסתכל על קבוע כעל תא בזיכרון בו אנו יכולים לאחסן מידע, אך האחסון נעשה באופן חד-פעמי. ברגע שקבענו את ערכו של הקבוע, לא ניתן יותר לשנות אותו לכל אורך התוכנית.

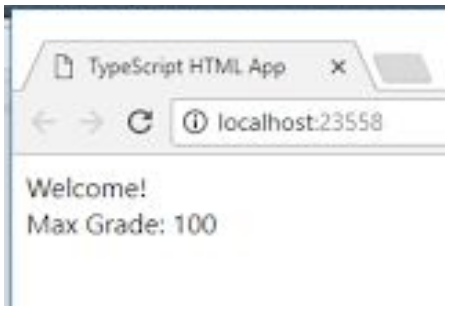
מספר כללים לגבי קבועים:

- חובה לתת לקבוע את ערכו מיד בשורת ההגדרה.
- לא ניתן לשנות ערך של קבוע לכל אורך התוכנית.
- השימוש בקבוע הינו בדיוק כמו שימוש במשתנה (למעט העובדה שלא ניתן לשנות אותו). בכדי שהיה קל לזהות במהלך התוכנית שמדובר בקבוע, נהוג ששמו יהיה באותיות גדולות ובין המילים יהיה קו תחתון (_) לדוגמאDAYS_IN_WEEK :

לשימוש בקבועים יתרונות רבים:

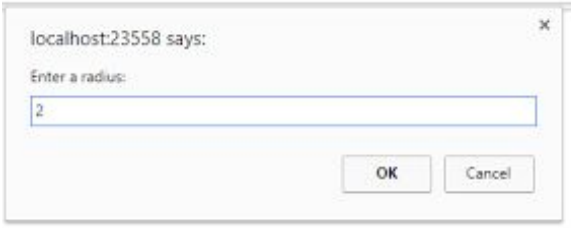
- קריאות ובהירות של התוכנית
- תחזוקה קלה –אם נשתמש בערך מסוים בכמה מקומות בתכנית, ובשלב מסויים נרצה שלנות את הערך, במקום לשנות את הערך בכל מקום בו השתמשנו באותו הערך, נצטרך רק לשנות את הערך המוצב לקבוע בשורת ההצהרה.

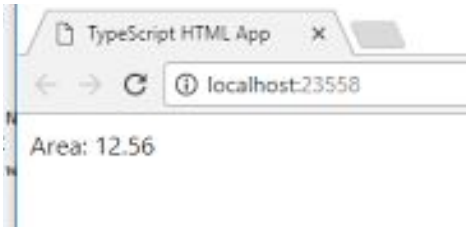
```
//13 - const

const MSG: string = "Welcome!";
//const header: string; // Error!!! Must enter initial value to a const.

document.write(MSG + "<br>");

//MSG = "Hello"; // Error!!! Can't change const.

const MAX_GRADE: number = 100;
document.write("Max Grade: " + MAX_GRADE);
```



תרגיל: קלוט מהמשתמש רדיוס, והצג את שטח המעגל. חובה ליצור את PI בתור קבוע.

```
//14 - Calculate Circle Area

const PI: number = 3.14;
let radius: number;
let area: number;



radius = Number(prompt("Enter a radius: "));
area = PI * radius * radius;
document.write("Area: " + area);
```





ביטוי בוליאני

- הגדרה: **ערך בוליאני** - ערך השווה ל-true או ל-false.(ברגע נתון יכול להיות או אמת או שקר)
- הגדרה: **אופרטורי השוואה** – Comparison Operators = אופרטור המחזיר ערך בוליאני.
- הגדרה: **ביטוי בוליאני** – ביטוי המשתמש לבדיקת תנאי מסויים על ידי שימוש באופרטורי ההשוואה ומחזיר ערך בוליאני.

אופרטורי השוואה

operator	is true when
a **==***b* equal	a is equal to b
a **<** *b* greater than	b is bigger than a
a **>** *b* smaller than	b is smaller than a
a **!=** *b* not equal	a is equal to b
a **<=** *b* greater or equal *a* **>=** *b* smaller or equal	b is bigger than a or equal b is smaller than a or equal

הטיפוס Boolean

- הגדרה: **ערך בוליאני**- ערך השווה ל-true או ל-false.(ברגע נתון יכול להיות או אמת או שקר)

בTS תחביר הגדרת משתנה בוליאני הוא באופן הבא:

let **שם-משתנה**: boolean = **ערך**;

כאשר הערכים היכולים להיות מוצבים לתוך המשתנה הבוליאני הם אחד מארבעה האופציות הבאות:

1) true
2) false
3) ביטוי בוליאני (מכיוון שתמיד יחזיר או false או true)
4) הצבה של משתנה בוליאני אחר לתוך המשתנה הבוליאני הנוכחי

תנאי בסיסי

Simple condition - execution of a block (chart segment) under certain conditions (the condition will return a Boolean value).

התנאי הוא כלי שמאפשר לנו לבצע פקודות מסוימות בקוד, בהתאם לתנאים שנקבע.

התחביר הוא כזה :

```
if (ערך בוליאני){
 //code to execute if true
}
else {
  //code to execute if false
}
```

תזכורת- ערך בוליאני עבור תנאי יכול להיות:

1) true

2) false

3) ביטוי בוליאני (מכיוון שתמיד יחזיר או false או true)

4) משתנה בוליאני

דוגמא לקוד מלא של תכנית המשתמשת בתנאי:

```
//simple if-else example
let a: number;

a = 10;

if (a > 10) {
        document.write("a is larger than 10.");
}
else {
        document.write("a is smaller or equal to 10.");
}
```

- כאשר בכל בלוק של התנאי ישנה רק פקודה אחת ניתן לוותר על הסוגריים, אבל תמיד עדיף להשאיר אותם כי כך הקוד קריא יותר.

לדוגמא:

```
//output by age
let age: number;

age = Number(prompt("Enter your age: "));

if (age > 20)
        document.write("Welcome!");

else
        document.write("You too young!");
```

לעיתים נרצה לבצע פקודה/ות במידה והתנאי מתקיים, אך לא לבצע שום פקודה במידה והתנאי אינו מתקיים.
במקרה כזה נוותר על החלק של ה else בתנאי if

לדוגמא:

```
//if without else
let a: number;

a = 20;

if (a > 10) {
	document.write("a is larger than 10<br>");
}

document.write("END");
```

תרגיל בנושא תנאים: צור שני מתנים מספריים, קלוט לתוכם קלט מהמשתמש, והצג את המספר הגדול מביניהם למסך.

```
//find max number
let a: number;
let b: number;

a = Number(prompt("Enter first number:"));
b = Number(prompt("Enter second number:"));

if (a > b) {
	document.write("Max: " + a);
}
else {
	document.write("Max: " + b);
}
```

תנאי מורכב

Complex conditions- The **&&** and **||** operators make up a complex Boolean expression that is created from two Boolean expressions

AND (&&) - operator that accepts two sub-conditions and returns true only if both are true, otherwise returns false.
The truth table of the && operator:

AND (&&)	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

OR (||) - operator that accepts two sub-conditions and returns true if at least one is equal to true, otherwise false returns.

The || operator's truth table:

OR (||)	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

NOT (!) - Operator that receives one sub-condition, and:

- If the condition is equal to true - returns false.
- If the condition is equal to false, true returns.

The truth table of the ! operator:

NOT(!)	TRUE	FALSE
	FALSE	TRUE

לדוגמא שימוש בתנאי מורכב:

```
let grade: number;
let factoredGrade: number;

grade = Number(prompt("Enter your grade: "));

if (grade >= 0 && grade <= 100) {

        if (grade >= 95) {
        factoredGrade = 100;
        }
        else {
        factoredGrade = grade + 5;
        }

        document.write("Grade after factor: " + factoredGrade);
}
else {
        document.write("Error!");
}
```

שילוב יותר מ2 תנאים כאשר ביניהם&& או ||:

- לאופרטור && קדימות על פני האופרטור || (כמו קדימות של כפל על-פני חיבור). ניתן לשנות את סדר הקדימויות ע"י שימוש בסוגריים (כמו בפעולות חשבון).
- אופן קביעת הערך של הביטוי (האם הוא מחזיר true או false) יהיה לפי הכלל הבא:
  ב && כל התנאים חייבים להתקיים כדי להחזיר true
  ב || מספיק שרק תנאי אחד יתקיים כדי שיחזור true