**DOCS – JavaScript 2022-12-26**

**Matrix - מטריצות**

מטריצה – היא מערך דו מימדי. כלומר , מערך של מערכים.

כל פריט במערך הינו מערך.

ריצה על מטריצה הינה ע"י שתי לולאות (לולאה מקוננת).

דוגמה לשימוש: טבלה, לוח כפל, משחקים, ציונים של שכבה המכילה מספר כיתות וכו. **­­­**

**שגיאות**

קיימים שלושה סוגים של שגיאות בפיתוח תוכנה:

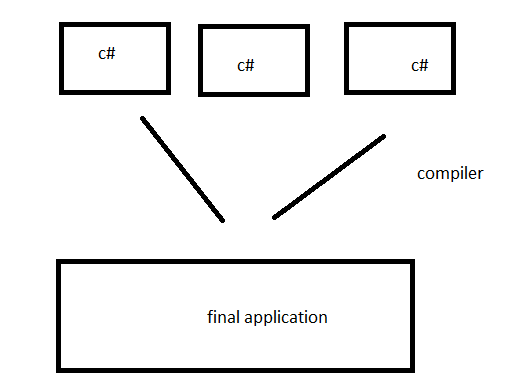
1. שגיאת קומפילציה – **הכי פחות מסוכנת**. היא לא נותנת לנו להמשיך עד שמתקנים את השגיאה; היא גם לא נותנת לנו להגיע לזמן ריצה.

קומפיילר (מהדר) – זה הוא מנגנון שהופך את הקוד שלנו לתוצר כלשהו, לדוגמה תכנית סופית ומציג שגיאות במידה והן קיימות.

ב JavaScript – אין קומפיילר. לכן אין שגיאות קומפילציה ב JS

ב TS, יש קומפיילר, לכן יש שגיאות קומפילציה.

שגיאת קומפילציה – זו היא כל שגיאה שמגנון ה compiler מגלה.



למשל – אם כתבנו תכנית, ובה יש פונקציה שמקבלת שני משתנים, אך בפועל העברנו אחד; התכנית תקרוס ולא תצא לפועל

**פתרון: קומפיילר – הוא יציג לנו שגיאות, אנחנו נראה אותן ונתקן אותן.**

1. שגיאת ריצה – התכנית תקרוס (או קורסת) בזמן הריצה ומפסיקה לרוץ.

אם זה קורה, אז רואים את הקריסה ב console

**פתרון: קומפיילר (אם יש) או מנגנון try and catch**

1. שגיאה לוגית – **הכי מסוכנת**. אף אחד לא מודע לקיומה. התכנית יכולה לצאת לדרך עם השגיאה הלוגית שלנו, כל שגיאה באלגוריתם שלא גרמה לקריסה.

פתרון : לא לעשות אותה 😊 , לבדוק את הקוד שלנו כמה שיותר, להריץ את התכנית שלנו/ את הפיצ'ר שבנינו, QA וכו וכו.

**Exception handling - טיפול בחריגות.**

**חריגה – שגיאת ריצה.**

ניתן לתפוס חריגות אם התרחשו , ולטפל בהן, בצורה כזו שהתכנית לא תקרוס.

זה מבוצע ע"י מנגון try-catch

כשאני כותב תכנית, אני לא יודע מה באמת המשתמש יכניס, איזה ערך יזין, ולכן אני מגן על עצמי בצורה הזו.

במידה וישנה חריגה בפונקציה, אז היא מחפשת את ה try catch ב scope של הפונקציה, אם היא לא מוצאת, היא הולכת למעלה, בפונקציה שעוטפת אותה וכך הלאה.

כמו בדוגמה הבאה : הקריסה הייתה בשורה 21, בתוך הפונקציה getPi, אבל אין שם מנגנון try and catch, אז היא יצאה למעלה לפונקציה displayPI, שבתוכה כן קיים מגנון כזה, שם הקוד היה עטוף ב try and catch, שם גם נתפסה השגיאה.

function displayPi(){

    try{

        const digits = +prompt("enter a number");

        const pi = getPi(digits);

        alert(pi);

    }

    catch(err){

        console.log(err.message);

    }finally{

        console.log('done')

    }

}

displayPi();

function getPi(digits){

    const originalPI = Math.PI;

    const customPI = originalPI.toPrecision(digits);

    return customPI;

}

**Throw**

אם הפונקציה מקבלת לסוגריים ארגומנטים המכילים מידע שגוי, אסור לה לבצע שום דבר פרט לזריקת חריגה ( throwing an exception ) בכדי לגרום לחריגה עבור הקוד שקרא לפונקציה (כי הוא לא היה בסדר עקב שליחת מידע שגוי).

אין להציג בפונקציה כזו הודעת שגיאה ( זו פונקציה חישובית, היא אמורה לבצע חישוב ולהחזיר אותו, היא לא אמורה להציג שגיאות – לכן יש תמיד לזרוק שגיאה).