# Javascript

שפת תכנות – כיום הנפוצה ביותר בעולם. (גם השפה היחידה שרצה בכל הדפדפנים).

שאגב, אפשר לפתח איתה אתרים, וגם אפליקציות למובייל.

משמעות השם: Java זו שפת תכנות אחרת (עבור backend) שיצאה לעולם בשנת 1995, בו"ז יצאה שפה הנקראת liveScript, אך בגלל שהמילה Java נהייתה לבאז וורד באותה תקופה, שינו את השם ל JavaScript. נוצרה לראשונה על ידי ברנדן איך.



script – זו שפה שמערכת כלשהי קוראת אותה שורה אחרי שורה ומבצעת אותה כמו "תסריט", לכן בסה"כ יש לנו javascript.

הודעה קופצת (מעין פופאפ):

alert("Write here something….")

הצגת משו בדף:

document.write("write something here + html…")

**משתנים:**

var firstName = "Erez" – since the begging of time (of javascript)

let lastName = "Yosepov" – since 2015

const dogsName = "Skype"

קלט מהמשתמש:

prompt("Enter some var here..")

+prompt("Enter some number here…")

משפטי תנאי:

if(condition) {  
do something….

}

if(condition){

do something..

} else{

do something..

}

מנגנון switch:

זוהי פקודה המאפשרת להחליף סדרת if-else המשווים ערך של משתנה בודד ולפי מבצעים סדרת פקודות.

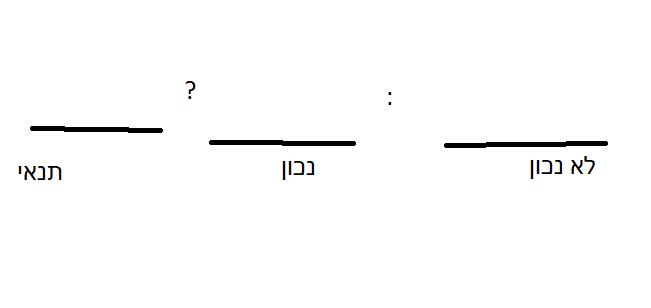
switch (var){

case someVariable: do something…;

break; - יוצא מהמנגנון switch

default: do something..; {ברירת המחדל – אם שום דבר לא מתאים.

משפט טרינרי –



משמאל – תנאי

סימן שאלה מסמנת סיום בדיקת התנאי.

מיד לאחר הסימן שאלה נשים את הפקודות לביצוע במידה והתנאי נכון.

אחרי הנקודותיים נשים את הפקודות שנרצה לבצע אם התנאי לא נכון.

|  |  |  |
| --- | --- | --- |
| אופרטור | דוגמה | משמעות |
| += | a += b | a = a + b |
| -= | a -= b | a = a - b |
| \*= | a \*= b | a = a \* b |
| /= | a /= b | a = a / b |
| %= | a %= b | a = a % b |
| \*\*= | a \*\*=b | a = a \*\* b |

|  |  |  |  |
| --- | --- | --- | --- |
| אופרטור | דוגמה | משמעות |  |
| ++ | a++ | a = a + 1 | קידום מאוחר |
| -- | a-- | a = a - 1 | חיסור מאוחר |
| ++ | ++a | a = a + 1 | קידום מוקדם |
| -- | --a | a = a - 1 | חיסור מוקדם |

**לולאות – while.**

לולאה המתבצעת כל עוד התנאי מתקיים (שווה true).

הבדיקה של התנאי מתבצעת לפני הכניסה ללולאה. לכן אם התנאי לא מתקיים כבר בהתחלה – אנו לא ניכנס ללואה.

תחביר:

while(condition) {

… do something

}

סדר פעולות:

1. בדיקת תנאי
2. אם התנאי מתקיים – ביצוע פקודות
3. חוזרים לבדיקת התנאי

לולאת do-while

לולאה דומה ל- while, אך הפוכה מבחינת סדר פעולות.

ראשית, מבצעת פקודות ולאחר מכן בודקת תנאי.

תחביר:

do {

..do something

} while(condition);

סדר פעולות:

1. ביצוע פקודות
2. בדיקת תנאי

אם תרצו אין זו אגדה (בלי קשר ללולאה).

בנוסף, אם תרצו לבצע פקודות כלשהן לפחות פעם אחת – לפני בדיקת תנאי כלשהו, תשתמשו בזה.

**לולאת for**

לולאה המכילה מספר חלקים: אתחול, תנאי, קידום.

תחביר:

for(let I = 1; I < 10; i++){

const num = 10;

alert(num + 1);

}

סדר פעולות:

1. אתחול.
2. תנאי
3. ביצוע פקודות
4. קידום
5. תנאי
6. ביצוע פקודות
7. קידום
8. תנאי
9. ביצוע פקודות
10. קידום

**break**

זו פקודה הגורמת ליציאה מיידית מהלולאה. התוכנית תמשיך בקוד מיד אחרי הלולאה.

while(condition){

if(condition) break;

document.write("hi");

}

**continue**

זו פקודה הגורמת למעבר מיידי לשלב הבא של הלולאה.

while(condition){

if(condition) continue;

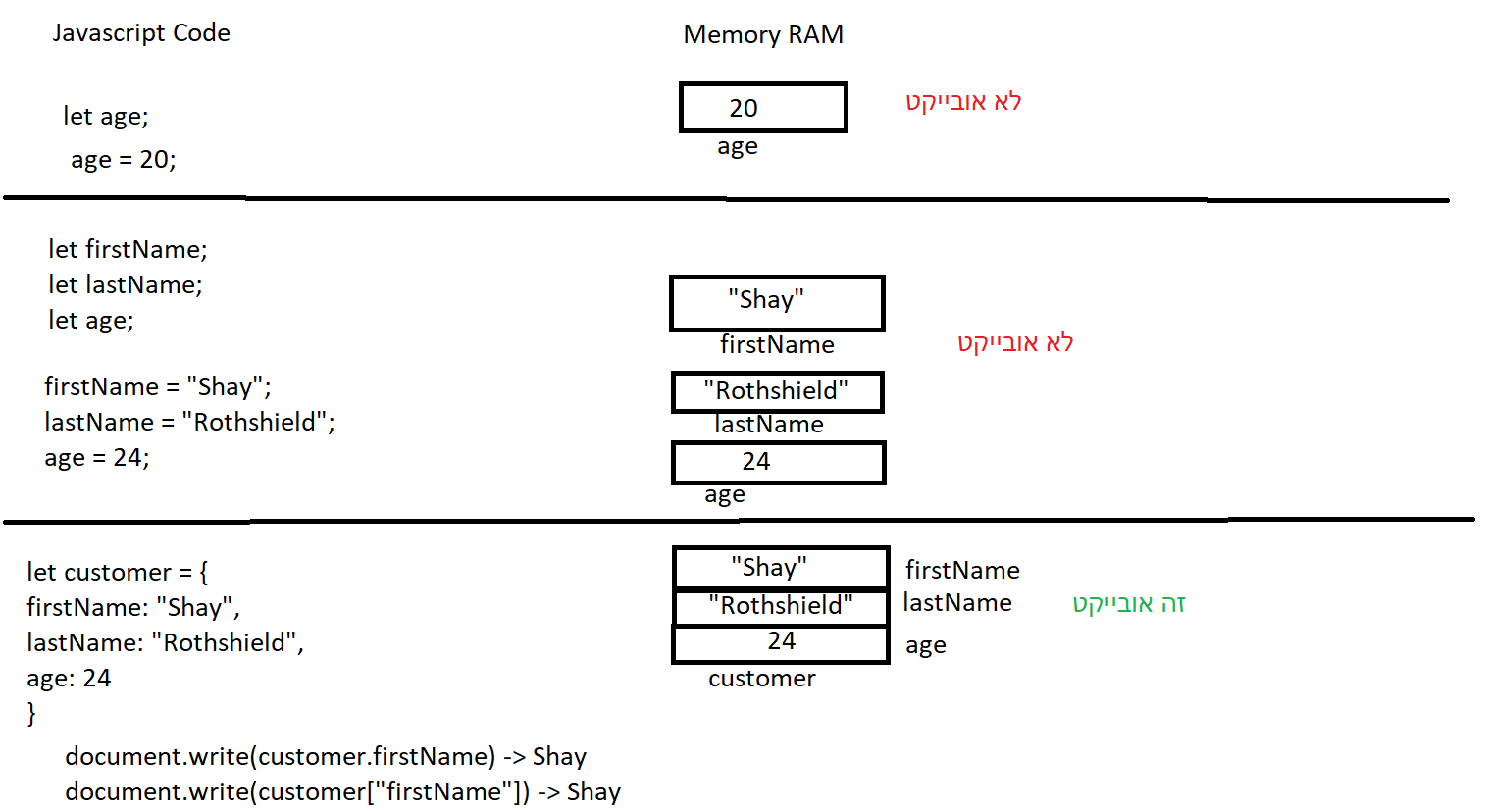
document.write("hi");

}

**אובייקטים – Objects**

אובייקט – Object זהו מבנה זיכרון הבנוי ממספר שדות.

לכל שדה יש שם. לאובייקט עצמו יש גם כן שם. ניתן להגיע לכל שדה ע"י שם האובייקט ואז נקודה ושם השדה.



אובייקט מוגדר ע"י סוגריים מסולסלים נקרא Literal Object.

קריאת מאפיין שלא קיים – תחזיר לנו את הערך undefined.

**Strings**

ישנם שלושה דרכים לכתוב בקוד מחרוזות:

1. אפשר להשתמש במרכאות כפולות " ".
2. אפשר להשתמש במרכאות יחידות ' '.
3. יש אפשרות נוספת, להשתמש בבאק טיקס back ticks ולהכניס לתוכן משתנים שיהוו חלק מהמחרוזת הכוללת בעזרת ${var}

לדוגמה: `This is some string with ${var} variable`

קיימת לולאה המסוגלת לרוץ על המאפיינים של האובייקט.

נקראת לולאת for-in תחביר:

for(const prop in object){

…do something

}

**מערכים – Arrays**

זהו רצף של תאים בזיכרון.

למערך יש שם.

לכל אחד מהתאים – אין שם.

לכל אחד מהתאים יש מיקום – index – היכן הוא נמצא ביחס לראשון.

המיקום של התא הראשון תמיד יהיה – 0.

המיקום של התא השני תמיד יהיה – 1.

המיקום של התא האחרון תמיד יהיה – אורך המערך – 1. array.length – 1

מספר התאים – length

ייעוד: להחזיק בזיכרון כמות גדולה של ערכים שלכולם יש מכנה משותף. (לדוגמה 100 ציונים של תלמידים).

תחביר:

const grades = [100,90,70,30……. X229E];

grades.length // 4

grades[2] // 70

grades[grades.length -1]

הוספה של פריט –

grades.push(76);

שינוי ערך בתא ספציפי

grades[1] = 20; // = [100,20,70,30……. X229E]

**לולאת for-of**

לולאה המסוגלת לרוץ על מערך (ועל אוספים אחרים).

תחביר:

for(const item of array) {

// do something

}

המשתנה בלולאה זהו כל אחד מפרטי האוסף. המשתנה הינו אינדקס.

הערות:

1. הלולאה יכולה לרוץ על מערך ובצורות משונות של אובייקטים.
2. היא תמיד רצה על כל המערך, ניתן לקרוא ל-break באמצע אם צריך.
3. משתנה הלולאה (item) הינו Read-only, לא ניתן לשנות אותו.

**יצירת מספר רנדומלי**

הפקודה Math.random() מחזירה מספר רנדומלי עשרוני.

המספר תמיד יהיה בין 0 ל-1. המספר יכול להיות 0, אך לעולמי עולמים לא יהיה 1.

אם ברצוננו טווח אחר, אנו יכולים לבצע פעולות מתמטיות על המספר, לדוגמה הכפלה, הוספה וכו'.

נוסחה להגרלת מספר רנדומלי שלם בין min ל-max כולל קצוות:

const num = Math.floor(Math.random() \* (max – min + 1)) + min;

לדוגמה:

min = 50 max = 70

const num = Math.floor(Math.random() \* (70 – 50 + 1)) + 50;

**String interpolation**

שיבוץ ערך בתוך מחרוזת. מתבצע ע"י הפורמט הבא:

const str = `some string ${value} continue string..`;

מאפשר גם לרדת שורה באמצע מחרוזת.

**Falsy**

**ערך השווה ל-false**

**אם נבדוק במשפט תנאי ערך זה, נקבל false.**

**לדוגמה: אם תוצאת בדיקה הינה false זה אומר שערך num נחשב falsy.**

**if(num){**

**//…**

**}**

**ישנם שישה ערכים הנחשבים Falsy כל היתר נחשבים ל-Truthy:**

* **False**
* **0**
* **"" או '' או ``**
* **null**
* **NaN**
* **undefined**

**Data – Types – טיפוס נתונים**

זוהי הגדרה המתארת כל ערך. לכל רך יש סוג ממנו הוא נולד.

לדוגמה, "hello" יהיה מסוג – string.

לדוגמה, 123 יהיה מסוג number.

ישנם מספר סוגים:

* string
* number
* Boolean
* object
* function
* undefined
* symbol

**Escaped Charters**

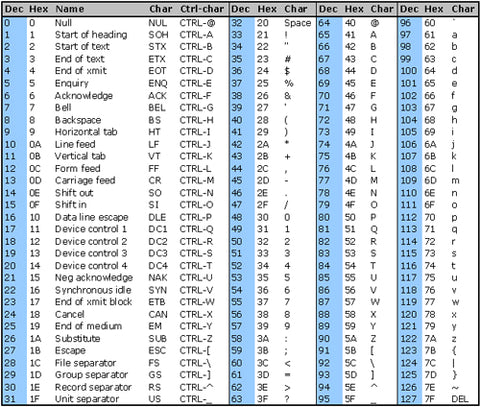
* \n – לרדת שורה
* \t – טאב
* \" – "
* \' – '
* \\ - \
* \r – חזרה לתחילת שורה – לא בג'אווה סקריפט.

**טבלת ASCII**

כל תו מיוצג במחשב על ידי מספר.

מקלדת אנגלית מכילה 128 תווים.

לכן טבלת ASCII מכילה רק 128 תווים.



כל האותיות הקטנות רציפות.

כל האותיות הגדולות רציפות.

כל הספרות רציפות.

כמו כן, קיימת טבלת ASCII רחבה המכילה 256 תווים.

טבלת Unicode

מכילה את רוב התווים בעולם, כולל עברית, ערבית, צרפתית וכו'...

מכילה 65,533 תווים!

**מטריצה – matrix**

מערך רגיל נחשב חד-מימדי.

מטריצה זה מערך של מערכים – שזה בעצם דו-מימדי.

**פונקציות – Functions**

פונקציה – קטע קוד המבצע מטלה (אחת בלבד) ספציפית ומוגדרת.

לקטע קוד זה אנו מגדירים שם.

במקומות שונים בקוד שלנו, אנו יכולים לקרוא לפונקציה והיא תתבצע.

אם נבנה פונקציה – ולא נקרא לה, היא לא תתבצע.

יתרונות:

1. שימוש בקוד חוזר – כתבנו קטע קוד פעם אחת והשתמשנו בו יותר מפעם אחת.
2. חלוקה של קוד לקטעים.

**Function arguments**

פונקציה יכולה לקבל ערך לצורך הפעילות שלה.

משתנה זה נקרא ארגומנט – argument.

הפקודה שקוראת לפונקציה היא זו ששולחת את הערך למשתנה.

המשתנה מוגדר בפונקציה בתוך סוגריים עגולים.

אם לא שולחים ערך למשתנה המוגדר – ערכו יהיה undefined.

**Variables lifetime**

זהו הזמן בו משתנה חיי. ממתי עד מתי הוא קיים בזיכרון וניתן לגישה.

משתנה המוגדר ישירות בתוך script נוצר ברגע שה-HTML נטען וקורא ל-script, ונהרס מהזיכרון ביציאה מהאתר (חיי לכל אורך הגלישה באתר).

משתנה המוגדר בתוך פונקציה – ארגומנט או משתנה שהגדרנו בקוד בתוך קוד הפונקציה נוצר ברגע שקוראים לפונקציה – ונהרס מהזיכרון ברגע שהפונקציה מסתיימת.

**Variable scope**

מהיכן ניתן לגשת למשתנה לצורך שימוש בו.

משתנים מוגדרים ע"י let ו- const.

* משתנה המוגדר ישירות ב-script – ניתן לשימוש רק אחרי שורת ההגדרה – בכל מקום.
* משתנה המוגדר בתוך בלוק כמו תנאי וכדומה, ניתן לשימוש אך ורק בתוך הבלוק עצמו בו הוא מוגדר – רק אחרי שורת ההגדרה.
* משתנה המוגדר בתוך פונקציה – ניתן לשימוש אך ורק בתוך הפונקציה, כמובן שרק אחרי שורת ההגדרה. אם הוא מוגדר בתוך בלוק הנמצא בתוך הפונקציה – הוא יהיה ניתן לשימוש רק בתוך אותו הבלוק.
* בתוך פונקציה ניתן להשתמש במשתנה שהוגדר בבלוק חיצוני לנו או ב-script הראשי.

לסיכום: משתנה המוגדר על ידי let או const ניתן לשימוש מהנוקדה בה הוא מוגדר ועד לסוף הבלוק בו הוא מוגדר.

משתנים המוגדרים על ידי var:

* מתבצע עבורו hoisting – הרמת המשתנה הכי למעלה שניתן – אם הוא בסקריפט הוא יעלה לראש הסקריפט, אם הוא בפונקציה – יעלה לראש הבלוק בפונקציה. כל זאת ללא האתחול שלו – הוא יראה לכם undefined.
* זו הסיבה שלא משתמשים עם var – אלא רק עם let ו const.

**החזרת ערך מהפונקציה**

פונקציה יכולה להחזיר ערך בודד, כמו מספר, מחרוזת, אובייקט, מערך וכו'...

פקודת החזרת הערך הינה return.

ברגע שיש פקודת return הפונקציה מיד מפסיקה לרוץ ומחזירה את הערך שיש מימין ל-return.

הערך המוחזר מהפונקציה יכול להתקבל למשתנה הנמצא משמאל לקריאה לפונקציה.

**חוק נכון ומוחלט:**

חובה לבצע הפרדה בין חישובים – לבין תצוגות חישובים.

פוקנציה שמבצעת חישוב – לעולם לא תציג את תוצאת החישוב או תבצע משהו אחר נוסף.

JSON – **J**ava**S**cript **O**bject **Notation**

פורמט להעברת או שמירת נתונים מהשרת לדפדפן -> לאתר.

הפורמט מתרגם ג'ווה-סקריפט לנוסח בו הכללים הבאים חלים:

1. הנתונים יהיו כתובים על ידי שם – וערך (key & values)
2. הנתונים מופרדים על ידי פסיק.
3. אובייקטים יהיו בתוך סוגריי מסולסלים.
4. מערכים יהיו בתוך סוגריים מרובעים.