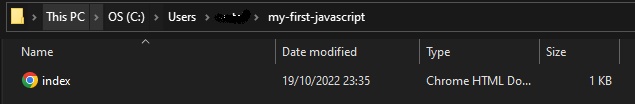
**EASS שיעור 12- JavaScript, Promises, CSS**

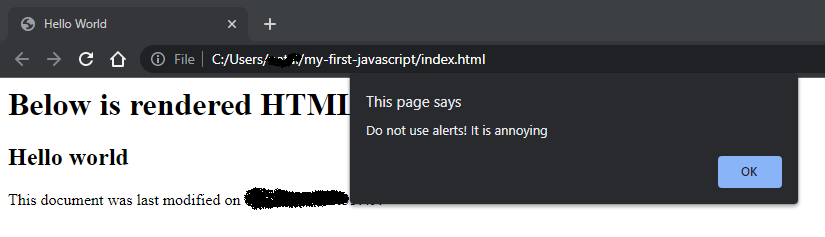
JavaScript:

* שפה סטנדרטית ל-UI.
* כדי שנוכל להשתמש בjavascript באתר, יש להטמיע את הקוד שלו בתוך קוד של HTML. ניתן לעשות זאת בעזרת תגית של HTML שנקראת <script>. דוגמה:

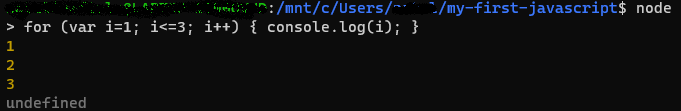






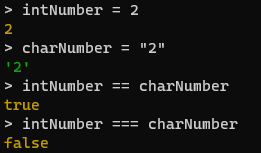
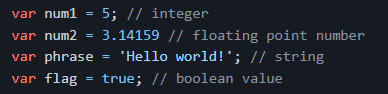
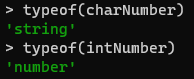
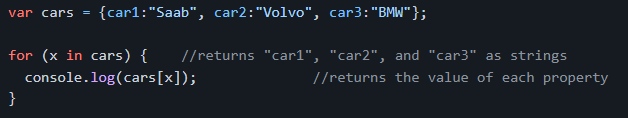
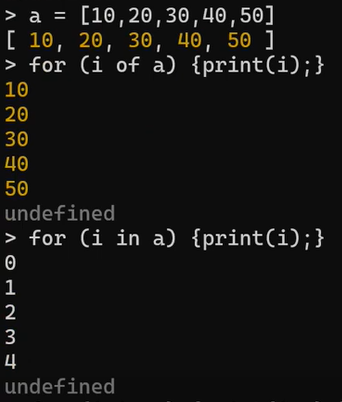
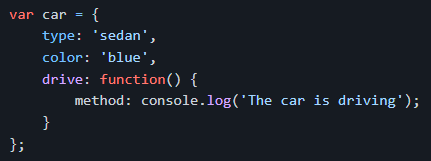
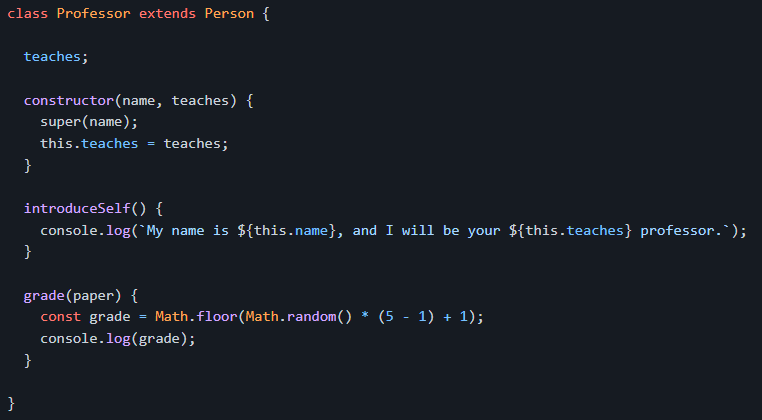


* בדומה לאיך שהשתמשנו ב-ipython כדי לכתוב ולקבל פלט בשפת python דרך WSL, נשתמש ב-node כדי לכתוב קוד ב-JavaScript.   
  תחילה נבצע את ההתקנה באמצעות פקודת sudo apt install nodejs או brew install node  
  ואז ניתן להשתמש בכלי node כך:



כדי לצאת מ-node נלחץ על ctrl+c במקלדת פעמיים או שנרשום .exit (נקודה ואז exit)

גם לולאת while פועלת בשפה זו כמו בשפת C או JAVA.

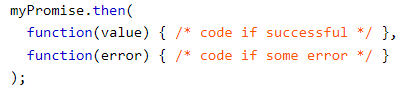
* השוואות בין משתנים בשפה עובדים בצורה הבאה:  
  אם יש == בין שני משתנים שאינם מאותו סוג, מתבצעת המרה (casting) של המשתנה השמאלי לסוג של המשתנה הימני  
  אם יש === בין שני המשתנים לא מתבצעת המרה של סוגי המשתנים.  
  
* נגדיר את המשתנים עם var שייקבע אוטומטית את סוג המשתנה בהתאם לערך שנכניס אליו:  
  
* בדיקת סוג משתנה:  
  
* לולאת foreach – עוברת על ערכים איטרביליים במבנה נתונים מסוג object (כמו רשימה, מערך וכו')  
    
  ההבדל בין of ל-in בלולאה מסוג זה:  
  
* Dictionary בשפה זו הוא JSON.  
  
* ניתן להגדיר מתודה כערך:  
  
* מחלקות ובנאים:  
  
* ירושה:  
  

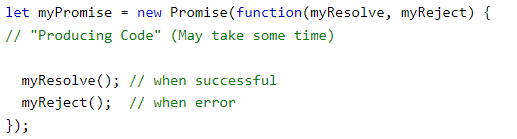
Promises:

אובייקט ש"מבטיח" להחזיר ערך בשלב מסויים.   
בנוי כקטע קוד שחייב להמתין לתוצאה של קטע קוד אסינכרוני (שיכול לקחת זמן).

ל-Promise יש שלושה מצבים:

1. Pending – מצב התחלתי, הפעולה לא הסתיימה.
2. Fulfilled – הפעולה הסתיימה וה-Promise החזיר ערך
3. Rejected – הפעולה נכשלה וה-Promise מחזיר את סיבת הכשלון.

Promise.then() – מקבלת שני ארגומנטים- אחד למקרה שהpromise יצליח ואחד למקרה שהpromise ייכשל.  


כאשר ניצור את אובייקט promise ונגדיר בו מהי הפעולה שיכולה לקחת זמן, ובהתאם לתוצאה שלה Promise.then() ייקבע אם נכשל או הצליח.  


מקור: <https://www.w3schools.com/js/js_promise.asp>

דוגמה נוספת:



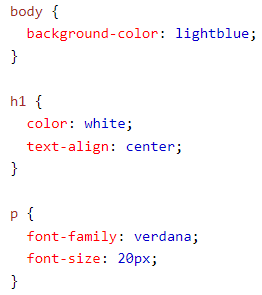
בשורה 3 הגדרנו את prom כבקשה שנשלחה לשרת. הזמן שייקח עד שתתקבל תשובה מהשרת אינו ידוע.   
לבנתיים, הפעולה בשורה 5 מתרחשת, למרות שהסטטוס של prom הוא עדיין pending , כי היא אינה תלויה בתשובה מהשרת.   
לעומתה, הפעולה בשורות 7 ו-8 ממתינה עד שתחזור תשובה ושהסטטוס של prom ישתנה ל- Fulfilled או Rejected , כי פעולה זו היא Promise.then() וכל שאר הקוד שאחריה גם ממתין עד שהיא תתממש.

הערה- ניתן לשרשר כמה then אחד אחרי השני שכל אחד ממתין לסיום פעולתו של הקודם.

CSS:

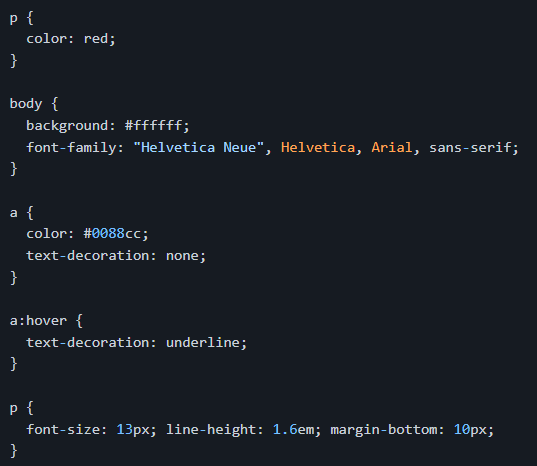
שפה שמעצבת ומגדירה מאפיינים עבור תגיות של HTML.

לדוגמה:



כל מה שמופיע בתוך תגית body יהיה עם רקע כחול בהיר, כל הטקסט שמופיע בתגית h1 יהיה לבן וממורכז, כל הטקסט שיופיע בתגית p יהיה בפונט verdana ובגודל 20 פיקסלים.

ניתן "לדרוס" או להוסיף בסעיף נפרד שינויים עבור אותה תגית. לדוגמה:

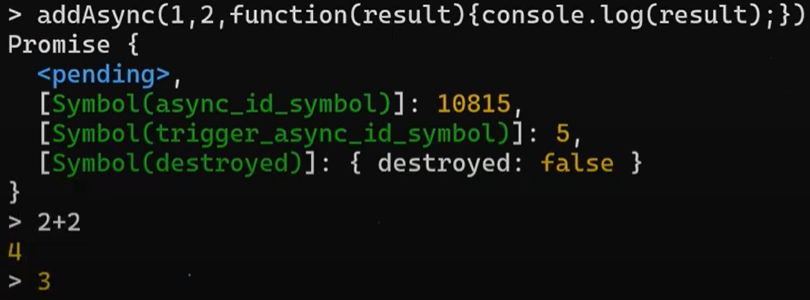


כאן נוסף לסעיף של תגית p שינוי נוסף בגודל הפונט.

פונקציות אסינכרוניות ב-JavaScript:

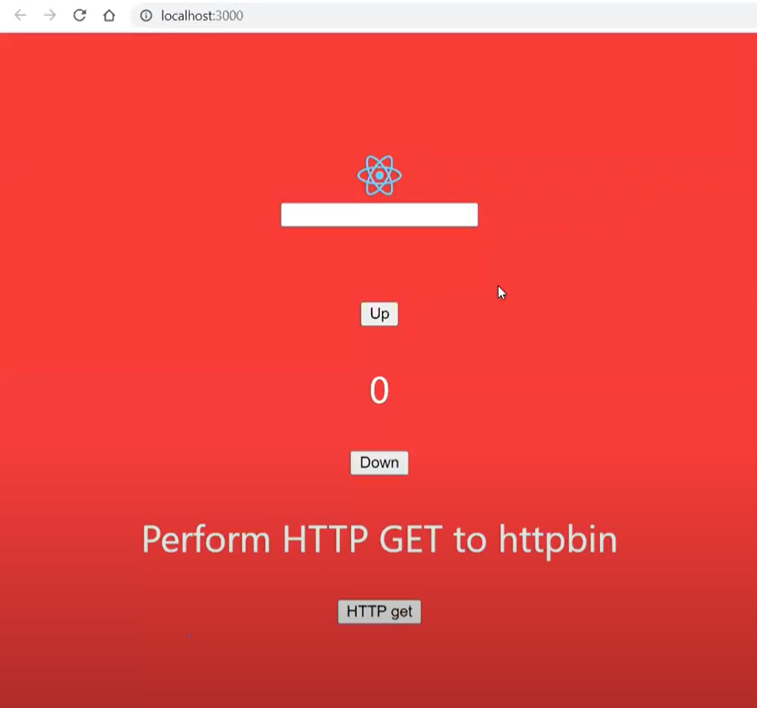


נגדיר פונקציה שמקבלת שני משתנים x,y ונגדיר שהיא מחזירה callback. הפונקציה מבצעת פעולת חיבור בין שני המשתנים, ממתינה שתי שניות וה-callback שלה הוא ערך משתנה תוצאת החיבור result.

  
מכיוון שהפונקציה היא אסינכרונית, בזמן שהיא רצה אנו יכולים לבצע עוד פעולות (כמו חיבור של 2+2) עד שתחזור התשובה שלה.

ReactJS:

ספרייה ב-JavaScript שמשמשת לכתיבת UI.

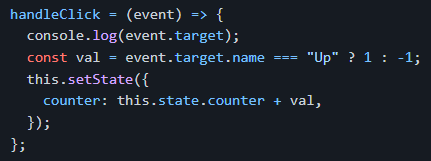
   
(הקוד המלא עבור דף זה נמצא במצגת שיעור בגיטהאב <https://github.com/EASS-HIT-2022/Part-A/blob/main/lectures/slides_part_a.md>)

נזמן את הספרייה בקובץ JavaScript:  
import React, { Component } from "react";



בחלק של render נכתוב את כל התגיות HTML שמייצרות את הדף, בתוך הסוגרים המסולסלים שיש בתגיות נכתוב קוד JavaScript, שבמקרה שלנו הוא מתודות שנקרא להן כשמתבצעת פעולה (כשלוחצים על כפתור, כשמכניסים טקסט לתיבת טקסט).

מעל לחלק של render , בתוך מחלקה שיורשת מ-Component של React, נכתוב את כל המתודות שלהן קוראים. לדוגמה המתודה שמתרחשת כשלוחצים על כפתור Up או כפתור Down:



ה-event הוא אובייקט. הוא מכיל בתוכו בין היתר את target שהוא אובייקט ה-HTML שקרא למתודה (במקרה זה הכפתורים).