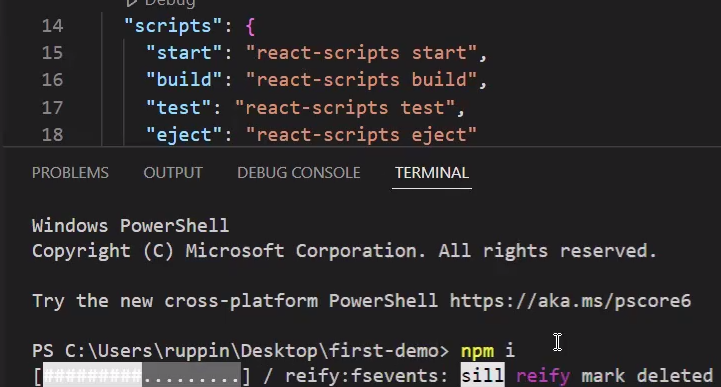
**שיעור 3** – VITE JSX

Clone from internet

URL

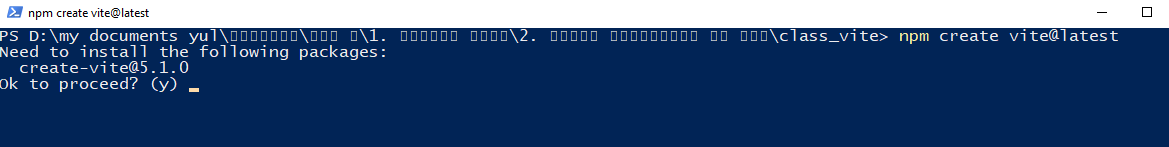
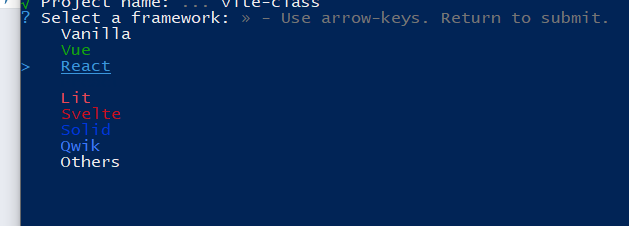
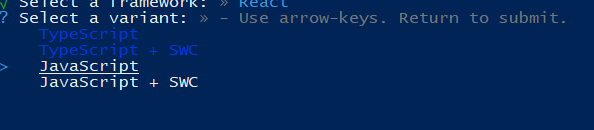
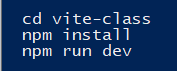
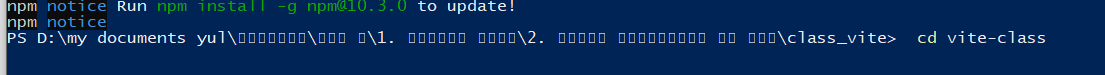
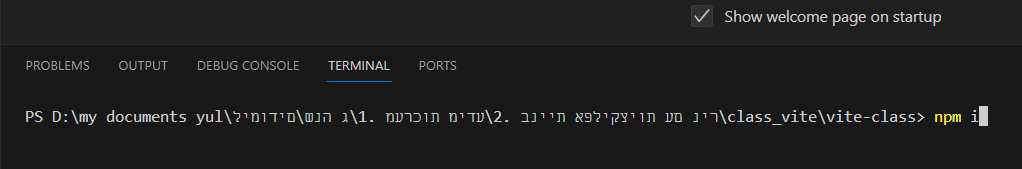
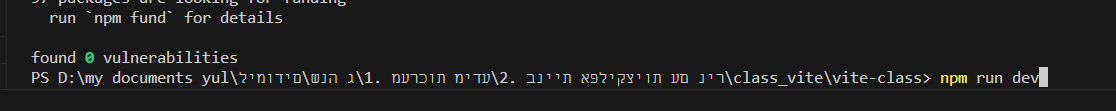
בהתקנה של התיקיות אחרי שכותבים code . עושים התקנה של התיקיות לתיקיה המסוימת (כל ה40 אלף קבצים של node models) באמצעות npm i

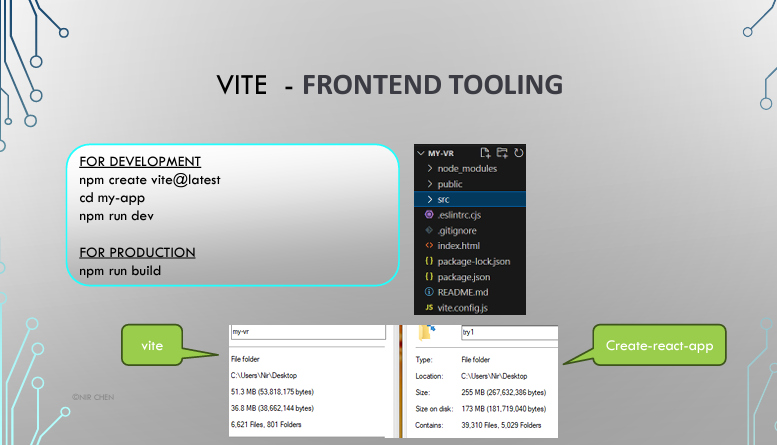


בטרמינל

Npm start זה להריץ.

Vite – פקודה שמייצרת פרויקטים של ריאקט וview ויותר מותאמים לclient side. זה מכווץ את הקבצים ככה שיש הרבה פחות קבצים.

1. פותחים תיקייה חדשה
2. עושים open powershall
3.  Npm create vite@latest
4. לוחצים y – כן.
5.  בוחרים שם לפרויקט למשל vite-demo
6. בוחרים שזה מסוג react עם החצים.
7. בוחרים JS השלישי
8. עכשיו מבצעים את שלושת השלבים 
9. 
10. פותחים ויזואל קוד
11. עושים npm i בתוך הויזואל כי זה יותר מהר(?)
12. עושים npm run dev 
13. מקבלים את הURL עצמו – הוא לא מריץ את זה. צריך לעשות קונטרול קליק על הקישור.



אפשר לעשות auto save כדי שזה תמיד יישמור שינויים.

אלמנט – חתיכת HTML סטטית שלא עושה כלום.

בתוך app.jsx  
מבחוץ לfunction APP   
  
const – אלמנט. נותנים להם שם שמתחיל בE. EHeader

כותבים HTML בפנים. למשל <div>

מסיימים בנקודה פסיק.  
האלמנט הזה יכול להכיל כמה HTML שרוצים, אבל כולם חייבים להיות תחת דיב אחד. תחת סוג אחד של HTML. עוטף אחד.  
shift alt F - מיישר.

הוספת אלמנט. כותבים עם סוגריים {EHeader}.

את כל האלמנטים עושים בתוך ספרייה של אלמנט  
shift alt a – לעשות הערות.

בתוך הספרייה (הוספתי ספרייה חדשה) אני עושה קובץ חדש   
EHeader.jsx – הסיומת של הX תיתן לי השלמות טובות יותר..   
צריך שהאלמנט יהיה public  
צריך לעשות לאלמנט export default EHeader – נתתי לו להיות בשימוש בקובץ אחר (דרך אחרת לעשות public).   
**צריך לכתוב export**לעשות import. כשכותבים EHeader הוא לפעמים עושה לבד את ההשלמה עם tab  
import x from ‘./ -כשעושים את הנקודה פסיק ההוא שואל מאיפה להביא את מה שאני רוצה ובוחרים. צריך לעשות פסיק כזה.

קומפוננט פונקציונלית

פונקציה שמחזירה HTML. היא צריכה לעשות return

היא יכולה לקבל props  
  
props – מה שהקומפוננטה מקבלת. לתוך הפרופס אפשר לשתול מידע. אני לא אשלח לו id,name,grade. שולחים פרופס נטו ואחרכך נכתוב:  
props.name  
A white background with black text

Description automatically generated

מעבירים את הפרמפטרים שרוצים באמצעות השם ו=  
name=”benny”

A close up of text

Description automatically generated

Function App זה בעצם קומפוננטה פונקציונלית כי היא מחזירה לי דברים על המסך מסוג HTML.

בפונקציות אפשר לעשות export בהתחלה. באלמנטים אפשר לעשות export default **רק בסוף**.  
צריך לעשות return עם **סוגריים עגולים**. והוא חייב להיות HTMLים שעטופים ב**דיב אחד** גדול.

סוגריים מסולסלים: כדי לגשת למשתנה מתוך הprops צריך להשתמש ב{}.

כדי להשתמש בCSS יש לנו כבר קובץ כזה app.css

className – הדרך לכתוב class (למשל בשביל להשתמש בזה בCSS).   
class זה משהו אחר בריאקט, מחלקה של JS. אז צריך להבחין בין הקלאס הזה לקלאס הזה.

כשבוחרים משתנה, למשל FCSTudent – לעשות טאב. ככה הוא יעשה אימפורט למעלה.  
כדי לשלוח דברים לשם, עושים כמו תכונות.   
  
כדי לשלוח מספר אני אעשה בלוק, כדי שזה יחזור להיות גאווה סקריפט.

בגלל שאני רוצה קומפוננטה שמעוצבת במיוחד רק אליה, אז לא נרצה קובץ CSS חיצוני. נרצה שהCSS יהיה מזוהה רק איתה.   
עושים style={} בתוך הדיב של הreturn  
ועושים {{}} – בלוק בתוך בלוק. כדי לעשות JSON ובתוכו מעצבים מה שרוצים. למשל border:  
לא צריך לכתוב px עם פסיקים ‘20px’ או לא לכתוב ופשוט לרשום בלי פסיק.  
אפשר לעשות גם IF  
IF(props.grade >=90){}  
let - נעשה משתנה לעשות שינויים בפרופס (כי פרופס אסור לגעת).  
gr = props.grade למעלה לפני הIF  
ואז עושים grade={gr}

עושים IF ובוחרים if condition

../ - אם זה ספרייה מעל אז עושים נקודות

Event – לעשות כפתורים  
<button> ובתוכו את השם.

להביא בוטסטראפ:

עושים הדבקה של לינק index html מתחת לtitle

עושים onclick={} ובפנים עושים btnLearn ולמעלה עושים פונקציה  
function btnLearn(){}

אני עושה gr+=2  
וזה ישנה את המסך רק פעם אחת. כי עשיתי return שמציג והוא הציג וסיים.

E.target.value

Input type text

Onchange={}

Function chnGend(e){}

הe מקבל מידע.

e. target.value – זה מה שמקבלים מהe. כלומר מה שמקבלים מהאינפוט הזה.

הוא יקרא לפונקציה לבד כשמישהו עושה change. זה אירוע event