21.01.2024

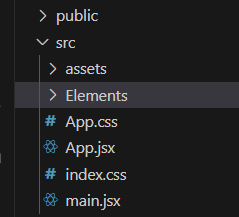
**שיעור 4**

VITE – מערכת שיודעת לאתחל פרויקטים גם של ריאקט וגם של VIEW ודברים אחרים. אין הבדל מריאקט.

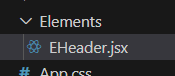
כשמתקינים VITE אם הוא נתקע אז עושים אנטר... או שעושים קונטרול C.

Shift alt חץ למטה – לשפכל שורה

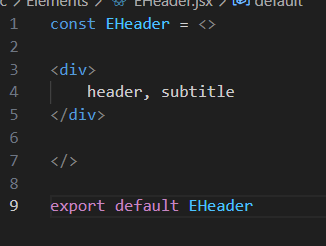
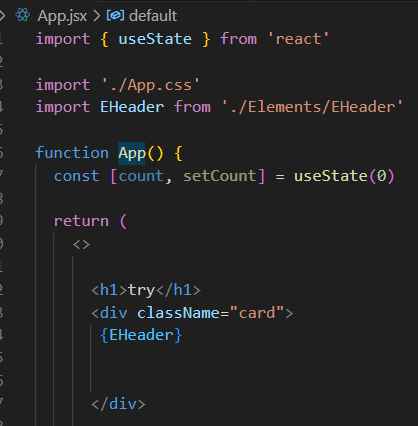
F12 – להיכנס לתוך הפנים של הפונקציה/אלמנט בקוד

<>, </> - **fragment** תג ריק, כי לא תמיד חייב DIV אם אין עליו סטייל, אז אפשר להחזיר בעצם רק תג ריק במקום דיב אחד שעוטף.

עושים תיקייה של אלמנטס.



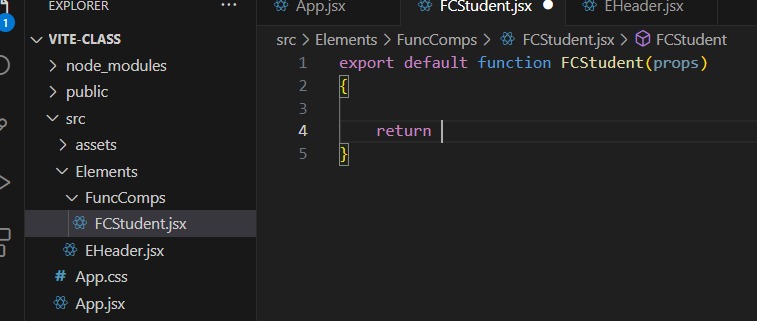
עושים אלמנט של HEADER – מתחיל בE גדולה.



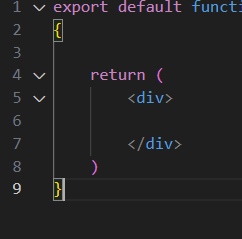
עושים export default

ואחרכך מביאים את זה לAPP עם לכתוב את הHEADER בתוך {} כאלה, ואז זה אוטומטי גם מביא import למעלה.

קומפוננטה פונקציונלית:



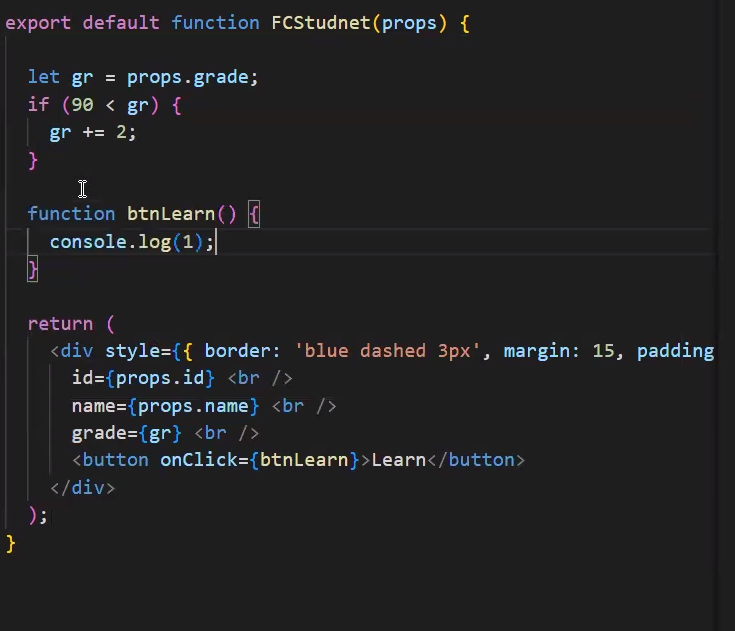
צריך להגיד לו מה להחזיר – בתוך הסוגריים של הריטורן

**למשל הדיב הזה.**

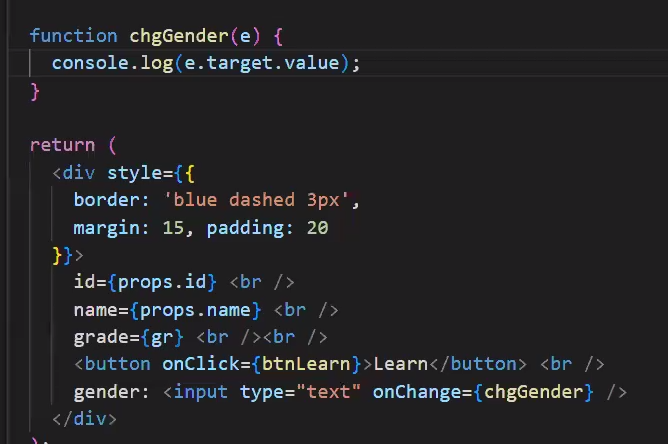
Props הוא read only, אז אם רוצים לעשות לו שינוי שמים את זה בתוך משתנה לוקאלי (לט) בודק באמצעות איפים ואז מחליף את המשתנה בהתאם.

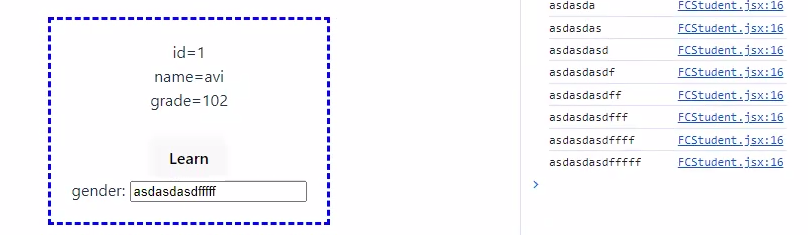


סיכום קטן של שיעור קודם:



**כדי לקבל מידע מINPUT – באמצעות e של event, דומה לגווי שנה שעברה**

**e.target.value**

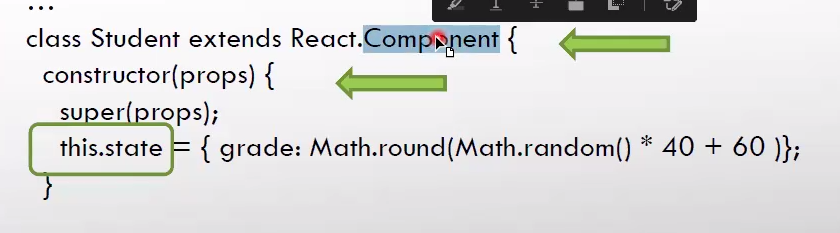
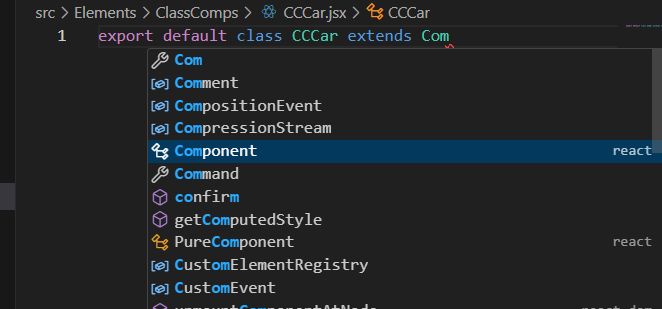
****

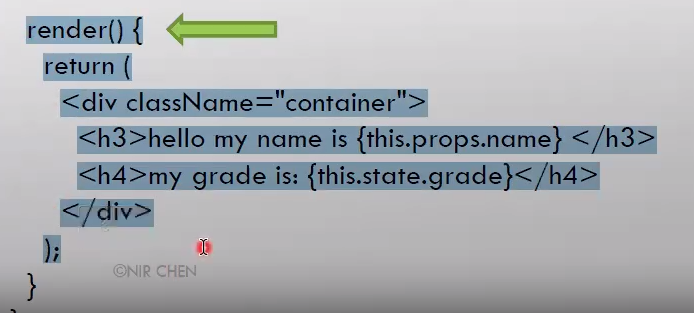
בדקה 18:00 ניר מסביר את התהליך והסיבה שאי אפשר לקרוא לפונקציית הרנדר עוד פעם בfunctional component – בגלל זה צריך קלאס קומפוננט – בדף הבא:

----

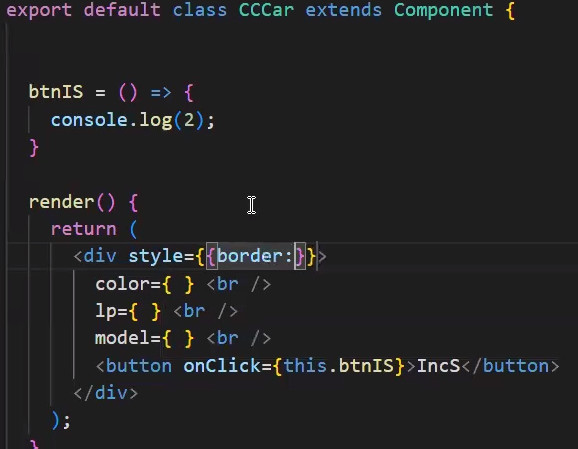
**Class component**

1. Extend component – זה יורש את הריאקט של קומפוננט. זה יורש במחלקה.

 בוחרים את הכתום.  ואז הוא מביא למעלה את ה

1. ועושים פונקציה בשם render שעושה ריטורן לדיב מסוים.

**בקלאס קומפוננט תמיד עושים פונקציות חץ!! וגם עובדים עם this.**

ככה btnIS עם הסוגריים ואז החץ.

This.btnIS – בקלאס צריך לקרוא להכל עם this

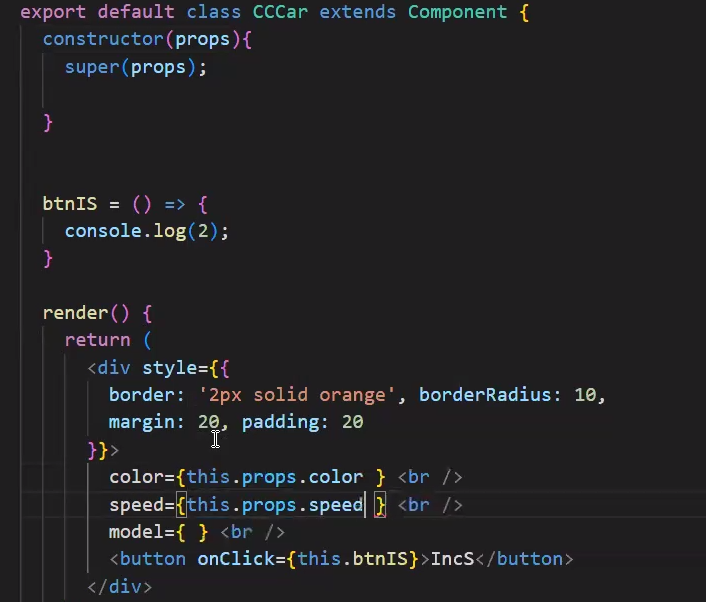
כדי לגשת לפרופס this.props.speed

כדי לשלוח פרופס – עושים קונסטרקטור: העברתי פרופס לקומפוננט, באמצעות בנאי.

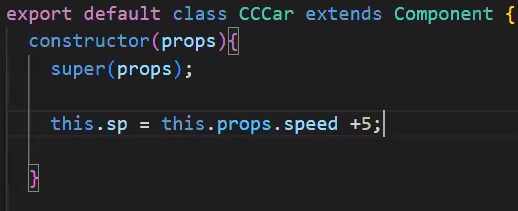
**שימוש בconstructor:**

בקונסטרקטור שולחים פרופס ופותחים {}

שולחים לו super(props) – ככה שולחים למשהו. כמו base



כדי לשתף מידע שהוא לא באמצעות let – למשל אני רוצה לשמור משתנה של speed ולעבוד עליו ולעשות לו תכונות – צריך להשתמש בשדה של מחלקה.

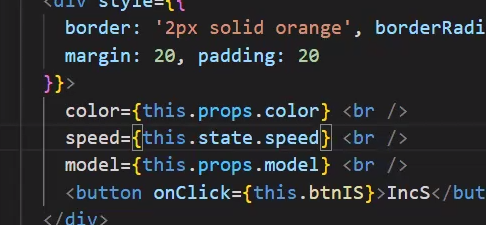
שדות:

This.sp = this.props.speed+5 – אני משנה שדות. **Sp יהיה השדה** שקוראים לו למטה this.sp

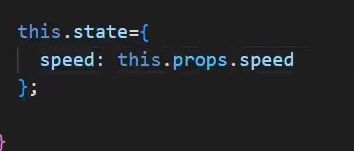


שדות הם הדרך שלי להביא משתנים שממשיכים לכל הפונקציה.

**State**

State – דרך לשנות ערכים במשתנים שמוצגים. setState מריצה את רנדר שוב – משהו שלא יכולנו לעשות קודם בפנקשיונל קומפוננט.

ניגשים לערך ההתחלתי באמצעות this.state.speed.

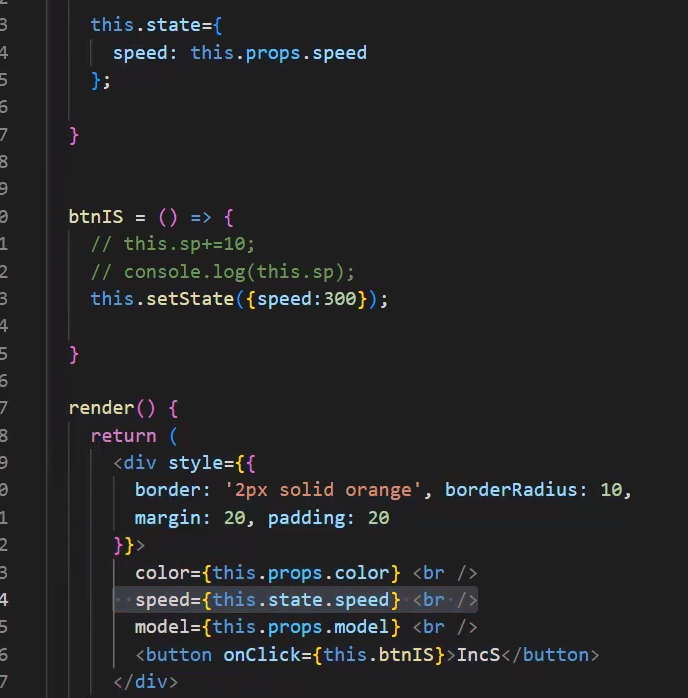
כותבים this.state="" בתוך הקלאס, פותחים סוגריים מסולסלים וכותבים בפנים **JSON** על הערכים שרוצים לאתחל. 

כדי לשנות את הערכים (אחרי האתחול) ובתוך הפונקציה של הonclick מפעילים פונקציה בשם setState

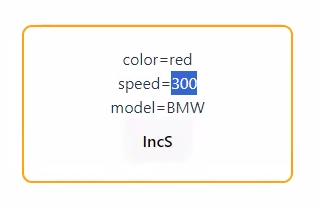
This.setState({speed:300}) – שולחים פרמטרים של JSON שרוצים שזה ישנה.

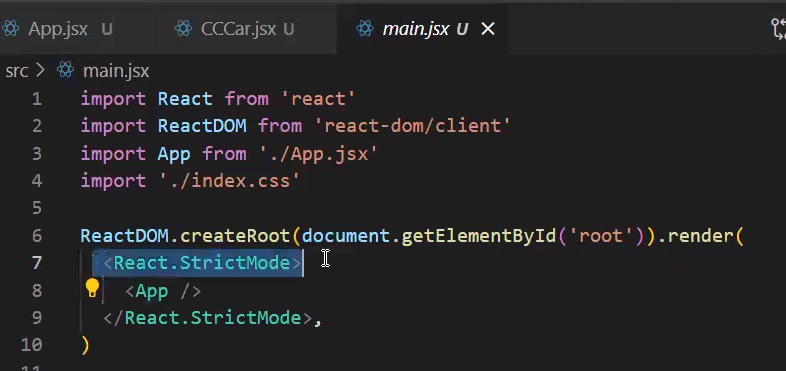


כלומר אני משנה את הסטייט speed למשהו אחר ואז זה **מפעיל את רנדר שוב**. אפשר גם סתם להכניס בסוגריים סטייט שהמצאתי שלא היה קיים קודם, לא חייב להגדיר אותם מראש.

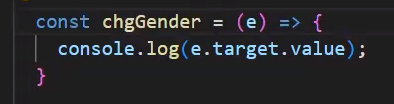
בהתחלה היה ערך התחלתי של 200 על המסך ליד speed.

ברגע שלחצתי על הכפתור, זה 1) שינה את הסטייט שנקרא "speed" ל-300.

2) גם עשה רנדר פעם נוספת בעצם.

מהmain מורידים strictmode כדי להוריד render שעושה פעמיים  


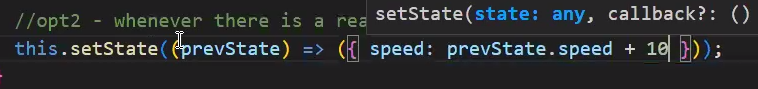
הפונקציה של הsetState סטייט **מפעילה רנדר נוסף בצורה אסינכרונית** וריאקט תריץ את הרנדר השני הזה כשנוח לה, יכול אפילו לקחת הרבה זמן.

כדי לעשות פונקצית חץ עושים const ברגיל:    
אבל בקלאס לא צריך const כדי לעשות פונקצית חץ

אנגלית

כדי לעשות שינויים, למשל לקחת ערך קודם ולהוסיף לו +10:

Previous state – כדי לקבל ערך קודם.



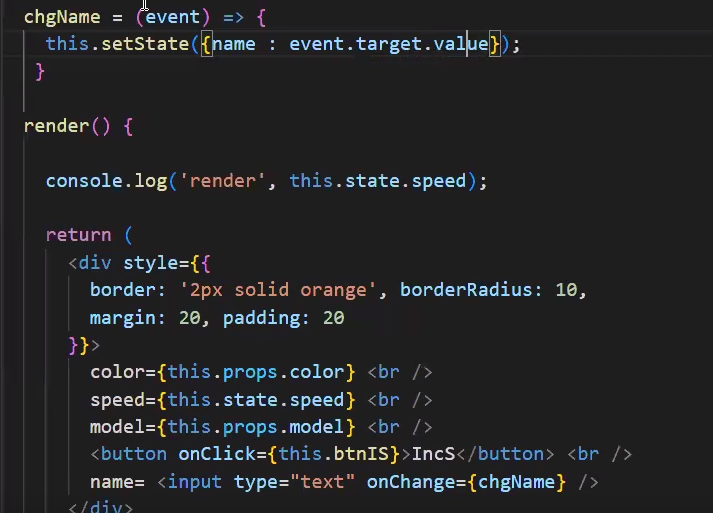
This.setState((prevS)=>({speed: prevS.speed+10}))

אם קיבלתי משהו לתוך פונקציית setState (בסוגריים) זה מקבל את זה כמו previous.

לרוב זה יעבוד גם בלי, אבל בתרגילים רציניים כבר צריך להשתמש ב-previous

אם אני רוצה תנאי – שואלים קודם האם השדה הוא עובר את ה250 מראש, ואם כן לעשות setState

פונקציה של שינוי שם:





Functional – פונקציה שמחזירה פעם אחת, אי אפשר לעשות לה רי-רנדר.

אם אני רוצה שהילד יעביר משהו לאבא (לא באמצעות פרופס שהאבא מעביר לבן), אם אני רוצה ש"הילדים" יעבירו בינהם משהו – זה חייב לעבור דרך האבא, ומעבירים לאבא באמצעות uplifting –

**Uplifting**: מעבירים פונקציה, מהאבא לילד ואז הילד יכול להפעיל פונקציה ולהחזיר לאבא. במקום להעביר לו 100 או avi מעבירים לו את שם הפונקציה(בלי הפעלה).

getData=(data)=>{}

לא לעשות alert.

הילד יעשה this.props.sendData() – וככה הילד יעביר מידע לאבא.