שיעור 31:

REACT:

ספרייה מאוד פופולארית.של UI
כתיבה יותר רצינית שמתוכננת לכך שהקוד יתרחב מאוד.
לאפליקציה יש סטייט של מצב המידע ומישהו צריך לנהל את המידע.
הרבה יותר מהירה ויעילה.
ספרייה מבוססת קומפננטות. - מחלק את הדף לחלקים.
קומפננטות יקבלו מידע.
בתוך קומפננטה יכולות להיות קומפוננטות אחרות.
נותן אפשרות לשינוי מהיר/
אנחנו נותנים לקומפוננטות שם וככה אנחנו יודעים שהם קומפוננטות וככה מקטינים את גודל הקוד.
רוב הקומפוננטות יהיו קומפוננטות לא מכולה.
ריאקט מג'נרט HTML.
אנחנו נכתוב HTML בתוך ה JS
בריאקט לא נבצע מניפולציה ישירות על הדום.
לקומפוננטה יהיו כמה אינסטנסים.
ללמוד את הדוקומנטציה של ריאקט.
יש כלי, REACT CREATE APP
נותן יכולות לייצר אפליקציה

Npx-create-react-app <app_name>

ככה מתקינים פרוייקט ריאאקט.
תמיד להריץ פקודות בטרמינל כשאתה בתוך התיקייה של האפליקציה.

Package.json

הקובץ הכי חשוב באפליקציה - כמו קונפיג.
אם אני מוריד אפליקציה ועושה npm install
המחשב מסתכל בפקדג' ג'ייסון ומוריד רק את מה שצריך לפי הפקדג גייסון.
פקג' לוק ג'ייסון דואג שהמחשב לא יוריד גרסאות חדשות לפקג'ים שקשורים אחד לשני.
גיט-איגנור אומר איזה קבצים לא להעלות לגיט.
דברים פרטיים או של אבטחה חייבים להוסיף לגיט איגנור.
ריאקט היא מנוע רינדור, מרנדר לתוך ה HTML עוד HTML עם כל הקומפוננטות

ID="ROOT"

דיב שבתוכו יש את הרוט הראשי של ריאקט. בלי זה ריאקט לא יעבוד.
הפקודה בטרמינל היא

Npm start

אפשר להריץ את הסקריפטים שבפקג גייסון תחת סקריפטס עם כפתור ימני וראן או בטרמינל.

SRC

היא התקייה היחידה שבה מייצרים קבצים.
לשם הריאקט מסתכל.
כל מה שאנחנו כותבים מתקמפל לדיב ID ROOT

הקומפננטה הראשית נקראת APP והיא מתרנדרת ל רוטו דיב, אנחנו לא נרנדר שום דבר אחר לרוט דיר. אנחנו נשים את כל שאר הקומפוננטות כבנים של הAPP,
מה שנכניס לקומפוננטה APP יהייה אינסטנסים של קומפוננטות אחרות שנגדיר אותם בקבצים אחרים.
הקובץ APP.JS יהייה הקובץ שבו נעשה את הדקלרציה של הקומפוננטה APP
כל קומפוננטה היא פונקציה - זאת הדרך הפשוטה. הפונקציה עושה RETURN ואחיה HTML

JSX
JS XML

קומפוננטות של ריאקט יחזירו JSX אלמנט - חתיכה של HTML.

Import react from react

הפקודה שחייבת להיות בכל דף קובץ של ריאקט.
סוגריים מסולסלות {}אפשר להכניס משהו מג'וואה סקריפט בתוך האלמנט JSX.
הדרך הקלה ביותר לייצר JS אלמנט היא על משתנה
רגיל הוא לא קומפוננטה והוא לא יעבוד כקומפוננטה JSX
תמיד שנכתוב JS בתוך הקומפוננטה הוא צריך לרנדר איזה JSX אלמנט.

<u>שיעור 32:</u>

המשך ריאקט:
בריטורן של הקומפוננטה תמיד יהייה JSX
לכל קומפוננטה יש תגית קומפוננטה ירוקה שמפעילה את הפונקציה של הקומפוננטה ומציגה את הריטורן של הקומפוננטה. כל פעם שקוראים לקומפוננטה עם התגית יוצאים אינסטנס של הקומפננטה.
כל קומפוננטה יודעת לקבל אובייקט ששמו הוא פרופס
כל המשתנים יהיו בפרופס.
אפשר להגדיר לקומפוננטה קוסטום אטריביוט בINLINE ב ריטורן של הקומפוננטה הגדולה יותר
האטריוביוט מגיע בתוך אובייקט כמשתנה. אפשר לעשות לו דיסטרקשן ואזוהשתמש בזה עם סוגריים מסולסלות בתוך הHTML ב JSX
בתוך הסוגריים המסולסלות אפשר לעשות || ואז משהו דיפולטיבי וככה לכסות את התחת למקרה שלא תקבל את האטריביוט.

INLINESTYLE

בריאקט מגיע {{ סטייל: }}
אפשר להעביר באובייקט את כל הסטייל כל עוד בונים בפונקציה הגדולה את האובייקט כמשתנה.
דיברנו על איך דורסים פרופרטיז דרך שנלחו באובייקט הפרופס ואצה לא רוצה להשתמש בהם.

IMPORT AND EXPORT:
אפשר לבצע אקספורט לפונקציות, משתנים, אובייקטים ומערכים זה מתבצע ע"י אקספורט סטייטמנט

Const param = “1”
Export param

בדף שאתה רוצה להשתמש במה שיצאת אתה צריך לעשות אימפורט

Import param from mypage.js

דיפולט אקספורט אפשר לבצע רק לאחד.
אקספורט רגיל אפשר לעשות הרבה

אם עושים דיפולט בדף שעושים אימפורט אפשר לקרוא למה שיצאת איך שאתה רוצה.
אפשר לעשות אימפורט לדפים שלמים עם שמים את הסיומת שלהם
מודול HEADER
אם עושים תיקייה לכל קומפוננטה ועושים אימפורט לתיקייה הוא הולך אוטומטית לקובץ INDEX.JS
בקלאס קומפוננט הפקודה RENDER()F היא מקבילה ל RENDER בקומפוננטה רגילה.

שיעור 33:

המשך ריאקט..
שני סוגים של קומפוננטות, פונקציונאלית וקלאס
אפשר להעביר פרופס מאב לבן..
ספרייה - מטריאל UI
להעביר אובייקט לקומפוננטה אפשר לעשות העתקה רדודה מהאובייקט שאתה רוצה להעביר ואז כבר אוטומטית יש לך דסטרקשן בתוך הקומפוננטה.

```
<product {...prop}/>
```

ככה

קומפוננטות סטייט:
אובייקט שיש מאחורי כל קומפוננטה שניתנת למניפולציה של מתכנת או יוזר, שיכול להתמודד עם דאתא שמשתנה.
בריאקט לא רצים על מערך
במקום זה משתמשים בפונקציית SETSTATE.
לא עושים פוש למערך בריאקט. בריאקט שינוי של הסטייט מצביע על רנדר מה שעושה DRAW אוטומטי.
לריאקט יש יכולת לדעת מה לרנדר. זה הכוח של ריאקט. יש דום וירטואלי, ריאקט מרנדר רק את מה שמשתנה.
זה הריאקט פייבר אנג'ין.
על כל איוונט בריאקטט אפשר לשים און צ'יינג'
בסטייט צריך לתת לכל ילד של קומפוננטה קיי

שיעור 34:

בריאקט לא משנה כמה פילטרים יהיו לנו תמיד נשתמש בפונקציה אחת עם &&
שבכל פילטר נשלח אלייה והפונקציה הזאת תמיד תחפש ב כל הדאטא.
פונקציה חיצונית של חישוב נבצע בתוך הקלאס קומפוננט, אם נכתוב אותו מחוץ לקלאס נהייה מחוץ לקונטנט.
נשאף לבצע כמה שפחות SETSTATE בגלל זה נשתמש בפונקציות חיצוניות שמשתמשות ב &&.
אפשר להעביר פונקציות לקומפוננטות בנים כפרופס. מעבירים את הפונקצייה בלי () הפונקצייה שנקרא לה בתוך קומפוננטת הבן תרוץ אצלך האבא ואם היא מריצה SETSTATE היא תריץ אצל האבא

שיעור 35:

המשך ריאקט…
כל קומפוננטה מורכבת מ3 חלקים, קונסטרקטור, לייב סייקל, ורנדר.
ה ANY ANY בלמעלה של הקלאס קומפוננט בטייפסקריפט ממצייצגים, הראשון הוא הפרופס והשני הוא הסטייט.
ועם טייפסקריפט אפשר להגדיר מה יהייה בסטטיט או בפרופס.

אפשר להגדיר את זה בראש הף עם INTERFACE וקלאס שבו האובייקט עם הטייפים

שיעור 36:

המשך ריאקט...
אם שמים וליו באינפוט בלי {} הוא הארד קודד בלי רינדור ראקט לא נותן רף לשנות את האינפוט, גם אם עשית עם{}

תמיד צריך להיות לנו סטייט ראשוני, למקרה שיש לנו קריאה מ API או משהו ויהייה רינדור ראשוני לפני שהסטייט של האבא יתרנדר.
אפשר לעשות במקרה ואנחנו לא מקבלים פרופס ריטורן מעל הריטורן של ה JSX.
ככה האפליקציה לא קורסת.
צריך להיזהר גם מלשים אנדיפיין או נל בתוך פילטרים.
טבלה אפשר להציג כולה בצורה דינאמית, הדבר הכי נכון לעשות היא לשלוח בפרופס גם HEADERS וגם DATA לשאר הטבלה
דרך מעניינת לעשות דיסטרקשן לאייטם במערך

Const [firstItem] = array

פעולות א-סינכרוניות בריאקט:

חייבים מידע התחלתי בסטייט, אם לא האפ תקרוס כי הקריאה עוד לא תסתיים.
אחת הפונקציות הכי חשובות בריאקט
קומפוננטת DIDMOUNT - componentDidMount
הרנדר קורה לפני המאונט. כי דידמאונט מופיע רק אחרי שיש משהו בדום ורק רק רק רק רק פעם אחת.
מתוך דיד מאונט בדרך כלל נריץ את הקריאות ל API.
נשתמש בחבילה שקוראים לה AXIOS (צריך להתקין אותה) אפשר גם להתקין ג'קווארי ולעשות עם אג'קס
AXIOSγGET(APIURL).THEN()

ראוטינג בין דפים שונים
כל כפתוק יחולל קומפוננטת דף חדשה. צריך להתקין עוד חבילה = ריאטק ראוטר דום
ריאקט מטיריאל דיזיין

שיעור 37:

המשך ריאקט..
קומפוננטות של ראוטינג - קומפננטת LINK
קומפוננטת ראוט יש גם PATH
קומפוננטה נוספת היא SWITCH כדי להחליף בין הראוטים
וקומפוננטת ראוטר שמחזיקה את היסטוריית הגלישה והיא עוטפת את כל האפליקציה, שאר הקומפוננטות לא יאבדו אם הם לא יהיו מתחת לקומפונטת ראוטר.
אפשר לשים קומפוננטת לינק בכל מקום, לא רק בNAVBAR, גם בתוך ראוט אחר.
קסטינג: בטייפסקריפט אפשר להגדיר במקום מסויים משתנה כטייפ מסויים ולא כהגדרה

VALUE AS STRING

לדוגמא

בקונסטרקטור של הקומפוננטות שנוצרת דינאמית מהראוט אפשר להשתמש ב
PROPS.MATCH.PARAMS
בראוט כדי להשתמש בPATH למשהו דינאמי אפשר לקרוא לו כמשתנה ככה
PATH="\COUNTRY\:CODE"
ואז נקבל את הנתונים ע"י PROPS.MATCH.PARAMS
אפשר גם בסוויטץ' להוסיף קומפוננטת ראואט עם פשאי של ** שיהייה קומפוננטות שגיאה.
יש בקומפוננטת ראוט גם אפשרות לכתוב EXECTLY ואז הוא לא יתקן ויסלח למשתמש על טעויות קטנות בכתובת.
חשוב שהראוט שגיאה יהייה אחרון.

סטייט מנג'מנט גלובאלי:

באפליקציה עם ROUTE אי אפשר לשמור מידע בדף ה APP ולכן נצטרך סטייט גלובאלי שעליו נשמור מידע.
בשביל זה יש את
REDUX:

.
רידקס היא ספרייה שמחברת ויוצרת סטייט גלובאאלי
צריך להתקין

REDUX
REACT-REDUX
REDUX-THUNCK

השחקנים הראשיים הם אפליקצייה עם ראוט וקומפוננטות.
המידע ישמר על אובייקט שקוראים לו STORE
יש סטור אחר באפליקציה.
כל התפקיד של הסטור זה לשמור מידע.
יש לסטור גם פונקציה שקוראים לה DISPATCH
המטרה של הדיספץ' היא לשגר אקשנים.
יש עוד יכולת שקוראים לה ACTIONS
כל אקשן זה פעולה.
רידקס זה מערכת סינכרונית ולכן בכל פעולה עם קריאה ל API יהיו 2 אקשנים.
כל אקשן זה פונקציה.
האקשנים בדרך כלל יצטרכו לשנות את הסטור, פונקציה של שינוי הסטור יהייה בדרך כלל REDUCER.
אחרי הרידוסר מתבצע CONNECT שהוא רנדר לקומפוננטות שמחוברות לרידוקס.

מעגל הרידקס
הסטייט מגדיר את ה UI הUI עושה דיספאץ' לאקשן. המידע של האקשן הולך לרידוסר. הרידוסר מעדכן את הסטור, שהסטור מכיל את הסטייט

שיעור 38:

המשך ריאקט רידוקס...
עדיף שלא ואבל אפשר לעשות תמיד STORE.DISPATCH)ACTION(S
ארכיטקטורה נכונה היא להעביר את הרידוקס רק לקומפוננטות הראשיות ב APP ואז לחלחל את הסטייט למטה לאט לאט, כדי למנוע רינדור יתר.

שיעור 39:

חזרה על ריאקט...

Async actions.

עד עכשיו עשינו פעולות סינכרוניות בלבד..
אם רוצים למחוק את הסטייט אז לא חייבים לשלוח בדיספאטש פיילואדץ רק טייפ יספיק.
ברגע שיש לנו רידוקס הקריאות לא חייבות להיות מתוך הקומפוננטה עד עכשיו.
לכן פעולת קריאה נבצע מתוך ACTIONץ וככה כל המעגל יקרה בתוך הרידוקס ורק אז יחזור לקומפוננטה.
כל אסינק אקשן בריאקט יהייה לו יותר מפעולה אחת.

עוד ספרייה

Redux thunk

עם פעולה MIDDLEWARE

שיעור 40:

המשך ריאקו, אם שמים תגית ריקה <> זה ריאקט פרגמנט ואז שמים שני אחים אחד ליד השני בלי קומפוננטה שעוטפת.

NODE JS

עד היום עברנו רק על צד לקוח על הדפדפן, היום נתחיל צד שרת.
בנוד כותבים קוד ב JS אבל לא מריצים בדפדפן, רץ ע"י מנוע V8 של כרום.
את הקוד של נוד נבצע על ה CLI - הטרמינלץ
נכתוב NODE ונריץ את הקוד.
הנוד נכתוב או על הטרמינל או שנכין קובץ הרצה ואז נפעיל אותו.

Node “filename.js”

יריץ את הקובץ עד שהוא יסתיים ואז הקובץ ימות ואם נרצה להריץ מחדש נצטרך להפעיל מחדש דרך הטרמינל.
נוד יריץ קוד של JS בכל מקום שתתקין אותו.
יש הבדל בין קבצים שמסתיימים לבד לבים קבצים שרצים בלי הפסקה ואתה צריך להרוג אותם עם קונטרול C.
ב נוד JS אתה יכול להריץ קובץ אחד כל פעם.
נעבוד הרבה עם IMPORT ו EXPORT רק עם שמות של REQUIRE ו השני...
תמיד נשתמש בנוד בהרבה פקג'ים ומודולים שלא שלנו.
כל מחשב שמחובר לאינטרנט עם ווב סרבר יכול להיות שרת.

הנוד הוא הווב סרבר אם מפעילים אותו.
לשרת יש יותר יכולות, דאטאבייסים, אבטחה, פייל סיסטם ופרוטוקולים יהיו בשרתים.
הוא סינגל טרדד JS
יש רק פרוסס אחד שמטפל בכל המשימות.
רק פעולה אחת בכל פעם.
נוד הוא א-סינכרוני, עם פעולות כמו קולבק בקריאות לשרתים כמו אג'קס.
מתודת FS פייל סיסטם יש הרבה מאוד פונקציות שימושיות של ראיה וכתיבה של קבצים וכו
fs.readfile("name, (err, data)
אם עושים ריד פייל א-סינכרוני חייבים לתת פונקציית קולבק, כמו בכל קריאה לשרת ב JS

UTF8
יוניקוד, אם אני מקבל באפרים במקום תשובה שאני רוצה לנסות לשים UTF8 כדי שיתרגם את זה.
נוד הוא OPEN SOURCE
קוד המקור הוא פתוח לכולם, אם מישהו רוצה לקחת חלק מהקוד מקור ולשנות אותו אז אפשר.
קוד סגור זה קופסא שחורה, קוד פתוח זה קופסא שקופה.
כל API שלנו יכלול
קוד שלנו, פלוס קוד של הקהילה פלוס קוד מקור של נוד.
בנוד כותבים מודולים. יחידה קטנה שעומדת לבד ולא תלויה בקוד אחר. עדיף כריוזיבל.
לכן כתיבה בנוד צריכה להיות נכונה.
תמיד נתחיל עם תיקייה למודול. שם שהוא טוב וברור ופונקציונאלי.
תמיד בכל תיקייה יהייה קובץ ראשי, עדיף אינדקס, הוא יהייה האנטרי פויינט ואותו נפעיל תמיד.
תמיד יהייה לנו
Package.json
תעודת הזהות של המודול.
בפקשג, ג'ייסון תמיד יהייה לנו השם, גרסא, דפנדנסיז (רק מה שהתקנו, לא מה שהוא הקור), סקריפטים ופקודות.
בתיקייה של המודול שיצרנו עושים NPM INIT וזה יוצר את הפקדג גייסון.
אם עושים INIT -Y א הוא לא שואל אותך שאלות.
אינסטול לדברים נבצע בתוך התיקייה של המודול.
כל מה שאתה מוריד יוצר תיקיית נוד מודולס, צריך להפוך אותו לגיט איגנור
Module.export
זה כמו אקספורט בריאקט
Module.export = { readJson };
מודול אקספורט אפשר לעשות מכל קובץ אבל אפ מישהו יעשה ריקוויר הוא יקבל את האינדקס

<u>שיעור 41:</u>

המשך נוד..
אם עושים אימפורט למודול הוא אוטומטי יקח את מה שנמצא בקובץ האנטרי.
במקום להוריד מהגיט נרצה לעשות INSTALL למודול או לפקדג' שלי
יש שרת גדול NPM JS. ממנו אנחנו מורידים את כל הספריות,
צריך יוזר כדי להעלות.
כל התקנה של פקדג הולכת לרג'יסטרי הגולובלי שלי.
יש לחברות גדולות רג'סטרי לוקאלי עם שרת משלהן.

נרצה גם להעלות דברים לרג'יסטרי
אפשר לעשות npm publish
וזה עולה לגלובל רג'יסטרי.
כשעושים לוגין עושים לרג'סטרי הגלובאלי
אחר כך עושים NPM PUBLISH
מה נמצא בנוד מודולז בREQIERE שלו לא צריך לכתוב את ה PATH. רק את השם.

אפליקציה צד שרת, צריכה לתת תשובות לצד לקוח.
צד לקוח, ריאקט, שרת, פוסטמן.
פונקציונאליות של שרת: ווב‌סרבר, עם פורט, ראוטינג, כיבה לקובץ.
גישה ל DB.
מנגנון אוטנתיקציה.,, למשל פספורט.
אג'קס או אקשיוס.מנגנון יקרא לAPI אחר.
יש מודול קוראים אותו אקספרס והוא מאוד חשוב
הוא מודול כדי לקבל ווב סרבר.

EXPRESS

פריימוורק מינימלי שהוא MIDELLWARES.

עוברת שכבות עד שמגיעה לבקשה של הקליינט.

הכל בנוי על מתודות, למשל GET או POST יביאו אותך למקומות שונים בשרת.
שרת צריך להאזין לפורט מסויים במכונה שלך..
מודול נוסף NODEMON
רק צרכי פיתוח, נותח לך יכולת לרפרש את השרת כל פעם שמשנים את הקוד.

10.103.50.112

<u>שיעור 42:</u>

המשך נוד JS
אנחנו תמיד נהייה עם שני פרוססים שרצים, בקאנד וקלליינט ושניהם צריכים לדבר דרך פרוטוקול HTTP.
תמיד נצטרך לעשות ריקווייר לכל מודול שנשתמש בו.
תמיד כדאי לבדוק את הרשיון של החבילה שאתה משתמש בה בקוד שלך.

אקספרס...
מי שמאזין על הפורט הוא המכונה, או המחשב או מכונה וירטואלית, THIRD PARTY
לכן אנחנו חייבים קולבק פאנקשן לפונקציה ליסטן.
אנחנו באקספרס כותבים קוד כדי להתאים את המטא דאטא של הבקשה לחתיכות קוד בשרת.

להוריד אפליקציה POSTMAN - איתה עושים טסטים לשרתים.

MIDDLEWARE

מידלוור זה פונקציה שיש לה גישה לריקווסט, לריספונס ויכולת לבצע נקסט.
חלק מרצף של פונקציות בכל קריאה.
פקדג'ז .
פוסט ריקווסט. למצבים שאנחנו רוצים לייצר או לשמור מידע,
המידע של POST לא עובד בקווארי פאראמז אלא בBODY.
כל שמירת מידע, בטח מידע גדול.

שיעור 43:

המשך אקספרס...
מתי נוד פחות טוב,
לא לעשות בלוקינג בבקשות, תמיד לעבוד א-סינק עם פרוססים גדולים, אם חייבים אותם עדיף לתת לפרוסס חיצוני או לפרטי שלישית.

DONTENV .ENV

יש דברים שצריך להצפין.
כשנצטרך לשמור על נתונים כמו סיסמאות או משהו, נרצה נגיד לתת APIKEY לכל משתמש
נרצה לכתוב קוד שהמפתח נמצא באופן מנותק מהקוד שבלי המפתח אי אפשר יהייה לראות את שאר הקוד.
מדובר בקובץ קונפיגורציה.

PROCESS.ENV

אם אתה עושה קונסול לוג מראה אובייקט גדול עם נתונים של מערכת ההפעלה.

Install env

require("dotenv"").config()

וקובץ

.env

שבו ממלאים ככה

PORT=4000
API_KEY=12345

מומלץ להוסיף אותו לgitignore

RES.STATUS(NUM)

אפשר לשלוט בהודעות השרת ששולחים למשתמש.
אפשר לשרשר לזה המשך של SEND

10.103.50.112

נעשה שיתוף בין פרונט אנד לשרת - נעבוד עם ריאקט

כשיש המון פאת'ים אפשריים בשרת נפצל אותם לכמה קבצים שונים ונחליט לאיזה פת' נעשה איזה מידלוור.
אפשר להגדיר ראוטר באפליקציה, לכל ראוטר יש סט של PATH שונים

את כל הקריאות של הראוט נמשוך מקבצים שונים וככה הדף יהייה יותר נקי ומסודר.

הפקודה היא EXPRESS.ROUTER

נפתח תיקייה של ROUTES
בתוכה נשים קובץAOTH.JS - כל הדברים שקשורים לאישורים יהיו בדף הזה
בדף של הראוט נעשה ריקוויר ל אקספרסס
ול EXPRESS.ROUTER()G

כל הקריאות בתוךהדף במקום לעשות API.GET
נעשה ROUTER.GET
נעשה מודול אקספורט לראוטר בסיום הדף

בריקווייר בדף הראשי ניתן לכל אחד מהראוטרים שם יוניקי

להוריד מודול WINSTON
מודול של לוגר
לקרוא את הדוקומנטציה של ווינסטון
לוגר אוטומטי
מייצרים בקובץ ביוטילז ועושים ריקוויר לקובץ שיצרנו ומוסיפים פקודות באנטריז השונים

!!!!!לא להעלות לוגים לגיט!!!!!!

לוגים זה א ב בשרת - ככה רואים מתי נפל השרת ולומדים

<u>שיעור 44:</u>

Hapi-joi-validation

מודול שעוזר לבצע ולידציה למה נכנס לאנטרי פוינט שלי בראוט.
כמו רג'קס.
מונע ממך לבצע מליון ולידציות ידניות.
אפשר לבצע ריקוווסו GET אל EALTHCHECK
כדי לראות אם הכל תקים והשרת למעלה.

<u>שיעור 45:</u>

SEED.DATA

להכניס מידע התחלתי לAPI
פעולות בין שרתים הם מאוד חשובות
להתאמן על קריאות AXIOS בין השרת שלי לשרתים אחרים

שילוב בין קליינט לשרת - שימוש בדומיינים שונים בין שני הפרוססים, דומיינין, סכמה או סאב דומיינים שונים

הבראוזר עצמו יכול לחסום קריאה, רק לקריאה. בין דומיינים שונים על אותו שרת.
צריך לעשות אנאייבל בשרת .
לפקודה שמבטלת וראים PERFLIGHT.

קרוס אוריג'ין קוראים לזה

אם התוכן הסטטי והשרת נמצאים על אותו ORIGIN אז לא צריך לעשות אינאבל לקרוס אוריג'ין

להתקין בשרת קורס CORS

APP.USE(CORS)

שיעור 46:

JWT
Json web token
ספרייה שנותנת אפשרות לג'נרט את התוקן המדובר, להצפפין מידע לתוך תוקן.
בשביל להצפין ולפענח חייבים מפתח סודי, אותו נשמור בENV ואף פעם לא נעלה אותו, כי עם המפתח אנשים יוכלו לפענח אותו.
עם החבילה הזאת אין צורך לשמור מידע אצלנו בשרת, האלגוריתם יוכל לעשות ולדציה.
אם יש שני שרתים שלשניהם יש את אותו מפתח, שרת אודח יכול להצפין ושרת שני לפענח בלי שהשרתים יצטרכו לדבר אחד עם השני

שתו מתודות חשובות של JWT

SIGN

ו

VERIFY

שזה אינקריפט ודיקריפט, רק שהוריפיי לא תמיד עושה דיקריפט לטוקן.

שיעור 47:

HOOKS

בפונקציות קומפוננטה נשתמש בUSESTATE של ריאקט
לייצק סטייט עם כל היכולות שלו, נכון לאותו דף קומפוננטה, עם לייפסייקל והכל
אפשר לייצר יותר מסטייט אאחד לכל קומפוננטה.

.**שיעור 48:**

תרגיל פיתוח:
דרישות לקוח:

אפליקציית שמנהלת לקוחות עם חשבונות בנק:
בבנק יש יוזר אחד, אדמין, מנהל יחיד, הוא נמצא מראש בDB.

.**שיעור 49:**

מאחורי ההוקס יש מערך גלובאלי שבו נשמרים כל השינויים
ולכן לא נשים IF לפני HOOKS
תמיד נשים את התנאים בתוך ההוקס, כדי לא לשבור את הרצף של הריצה של והשמירה של ההיסטוריה של השינויים בסטייט של ההוקס.

MYSQL

נתחיל לדבר על DB.
עד עכשיו עבדנו בשרת עם קבצים ומשתנים. זה לא אופטימלי ולכן נשתמש באפליקציית DATABEST.
פרוסס שונה לחוטין שעובד על המחשב שמהווה בסיס נתונים עבור השרת.
השרת והDB יתממשקו והמידע יהייה ממויין ב DB
אפשר לעבוד מול יותק מפרוסס אחד של DB.
מייסקל מותאים לדאטא רלציוני, שטוח. בניגוד למידע נסטד כמו ג'ייסון.
בין השרת ל DB יהייה API ונצטרך להתקין משהו בשרת כדי לדבר עם ה DB.
פורט דיפולטיבי של MYSQL הוא 3306.
אפליקציות יעודיות לניהול הDB

RDBMS

גורמת לנו לוותר על השרת לחלוטין, אבל זה אפליקציה יעודית.

זה שפה SQL
Structure query languege

אנחנו מבצעים בקשות לדאתא בייס בשפה שלו.
מייסקל הוא OPEN SOURCE
שפה חינמי
בחירת DB הוא אחד הדברים החשובים שבונים אפליקציה וצריך לחשוב טוב על טיב הדאתא והפעולות שלך לפני שמחליטים על מה לעבוד.

הכל מתמרכז בעבודה מול DB זה המהירות.
מיסקל הוא מולטי ת'רד, כל קריאה פותחת ת'רד.
טרמינולוגיה:
הרכיב העיקרי DATABASE - אפשר לייצר כמה DB ביחד. כל DB יקרא סכמה.
בתוך DB יש TABLES - (בDB לא רצציוני יקרא קולקשן) טבלה היא יישות שמייצגת ENTITY.
בחלק הבא - COLUMMS - טורים - כל ה "KEYS" של הטבלה.
החלק הכי חשוב ROW - הדאטא עצמו. כל מוצר הוא שורה ב DB.
צריך להיזהר בDB שאנחנו לא מכניסים נתונים כפולים.
החלק הבא - "KEYS" יש פרימרי קיי ויש פורן קיי.
פרימרי קיי הוא הקי הראשי, חשוב לתת את זה בכל טבלה כדי שיהייה קל להגיע לשורה הנכונה.
הפריימרי קיי יכול להיות מהלחם של שני קייז שונים.
פורן קיי יכול להיות מה שמקשר בין טבלאות שונות

.**שיעור 50:**

המשך SQL...
הראיון בשאילתות הוא להביא את רוב החישובים הקשים ל DB ולא לשרת, לשרת יש יותר כוח, הוא קרוב יותר למידע והוא מולטי ט'רדד.

פונקציית COUNT במייסקל סופר את הורות שיש לי, שמים אותה ב SELECT.
פונקציית DISTINCT, נראה רק פעם אחת מכל פילד.
אפשר לשרשר פונקציות כמו
count(distinct(first name)) as first name distict

אפשר לשרשר שדות פונקציית CONCAT לשדה אחד.

יש אפשרות לעשות גם פעולות מתמטיות.

בתוך הSELECT.

פונקציות מתמטיות נוספות
AVG SUM MAX MIN

שאילתה עם SORTING,
אחרי התנאי של WHERE יבוא פקודה ORDER BY..
הדיפולט של אורדר ביי הוא מקטן לגדול בשביל להפוך ניתן פקודה DESC.
אפשר לתת גם ASC

המילה LIKE כתנאי עם סטרינג למשל "&K& אחריה משמשת כמו INCLUDE פונקציה.

כשבודקים אם משהו קיים עושים IS NULL או IS NOT NULL

.**שיעור 51:**

המשך SQL

נלמד להתחבר למייסקל עם מודול מהנוד. נבנה שרת מודולרי.

לקרוא על ENTITY_MODEL_ENTITY.

כשCOLUM מטבלהה אחת הוא חייב FK ששל קולום מטבלה אחרת. אז הם לא יכול להיות שיש את ערך בעמודה אחת והוא לא קיים בטבלה אחרת. כמו כן, כל טבלה חייבת פריימרי קי.

פורן קיי יכול להיות מקושר רק לקולום אחד בטבלה אחרת.

טבלה עם הרבה עמודות היא משהו שננסה להימנע ממנו, ננסה להפריד בין טבלאות ולקשר את העניין כן FK.

פונקציית JOIN
תעזור לנו מתמטית לצרף בין שורות מטבלה מסויימת לשורות בטבלה אחרת, ג'וין מתבסס על הפורן קי.
טייפ אחד ההוא INNERJOIN בתוכו יש LEFT RIGHT או OUTER.

אינרג'יון נותן את החיתוך בין שני שאילתות מטבלאות שונות.

לפטג'וין או רייטג'יון נותן את כל המידע בטבלה שבחרת לפי סדר הכתובה, הטבלה השנייה תציג את רק את מה שבחיתוך.

אווטר ג'ון מציגה את כל הדאטא ומחברת את מה שקיים בתוך החיתוך.

אם נרצה לבנות DB עם רבים-רבים להשתמש בטבלה מגשרת זה פיתרון טוב.

את התנאי של הJOIN עושים עם ON ואז עושים את השדה שווה לשדה מהטבלה השנייה

Select * from a innerjoin b on id = product_id.

או

On a.fk = b.pk

בDOT<NET יש ENTUTY FRAMWORK - עוזר להבין איך DB עובד אם אתה ממפה את זה ככה.

בסט פרקטיס אם עושים ג'וין הוא לבקש בסלקט את הערכים מהטבלה הנטיבית.

חיבור של מיסקל לנוד.

לחבילה קוראים

MYSQL2

חיבור של קוואר‍י ל DB תהייה פעולה א-סינכרונית.

.**<u>שיעור 52:</u>**

בחיבור למיסקל עם MYSQL2 נשתמש ב POOL CONECTION
כל קונקשן שיפתח יבצע את הפעולה שלו ויעבוד ל POOL
אם אחד עסוק, קונשקשן אחר יכול לבצע את הפעולות.
אפשר לעבוד רק עד 10 קונקשן..
מנוהל שקוף לנו,
אנחנו עובדים עם .POOL PROMISE

.**שיעור 53:**

Node - mysql - pool

כשיוצרים טבלה במיסקל חשוב ללחוץ לעמודת ה ID על כפתור ה AI

AUTO INCREMENT

התוכנה תשלים אוטומטי , תג'נרט שדה ריק באותה עמודה.
לסטרינגים נרצה להשתמש ב VARCHAR45.

לעבור ע דאטאטייפס בנוגע ל MYSQL.
כפתור NN עושה ולידציה שכשמכניסים מידע השדה הנוכחי לא נשאר ריק.

לפני שמכניסים סיסמא לDB צריך לעשות לו HASH

ללמוד BCRYPTJS

.**שיעור 55:**

עוד אופציה לאוטנטיקציה.
שינוי עץ של הקומפוננטות, קומפוננטת אבא שבה יש את האוטנתיקציה.
לקרוא יל SUSPANCE בריאקט.

HIGH ORDER FUNCTION

פונקציה שמקבלת פונקציה, כמו קולבק.
יש גם היי אורדר קומפוננט שאפשר לעשות משהו דומה...אתה נותן לקומפוננטה הגבוהה את קומפוננטה בת.
אם התנאי עובד עוברים לקומפוננטה בת, אם לא אז זורק אותך החוצה.
אפשר להגדיר בטייפסקריפט קונפיג בריאקט BASEURL, מאיפה נתחיל את האימפורט שלנו לכל מה שאנחנו מביאים ב IMPORTT

פרוייקט 3:

.**שיעור 56:**

ANGULAR:

פריימוורק של גוגל,
בחיפושים על אנגולר לכתוב אנגולר ולא אנגולר ג'י.אס

אנגולר לא יצבה, אנחנו עובדים עם גרסאות 2 ומעלה.
לא דומה לריאקט.
יש מודולים מובנים של אנגולר שכבר נמצאים ושאין את זה בריאקט.
יש מודולים שאפשר לעבוד איתם רק באנגולר.

בניגוד לריאקט שמג'נרטים HTML עם הJS באנגולר אתה מרחיב את היכולות של ה HTML - אתה תכתוב דברים ב HTML.
בניגוד לריאקט שהיא היי פרפורמנס בדום וב UI ב אנגולר עושים מערכת לוגית גדולה יותר.
אם רושמים משהו ב TS זה משתנה ב HTML ולהיפך.
יש אפשרות לעשות COSTUM לדיירקטיב באנגולר, להוסיף לוגיקה לפרופרטי שאתה מוסיף ב HTML, יש גם דיירקטיב בילט אין.
יש סטייט גלובאלי גם באנגולר, STATERX.
בקומפוננטת אנגולר הטמפלייט יהייה בדרך כלל HTML, לכתוב טמפלייט מסודר, לא אינליין.
המטאדאתא הוא אובייקט ג'ייסון בתוך דקורייטור של קומפוננט.
יש אפשרות ב CLI לג'נרט קומפוננטה

ללמוד שיעור הבא על טייסקריפט קונפיג

עם פוליפילז אפשר לתת לבראוזרים ישנים לתת יכולות שאין להם תמיכה.

שיעור 57:

המשך אנגור - מחולק למודולים, כתוב כ @NGMODULE
שלושת הקריאות במודול, דקלריישן, אימפורט ובוטסטראפ.
יש גם פרוביידלרז.
בתוך מודולים יש קומפוננטות. @COMPONENT.
בתוך הקומפוננטה יש סלקטור - שם הקומפוננטה, HTML TAG.
יש גם TAMPLATEURL - יהייה כבר קובץ HTML.
יש גם STYLEURL - מערך שאפר להגדיר בו CSS.

אחרי זה אנחנו כותבים לכל קומפוננטה CLASS. מי שמייצג את הקומפוננטה.
בקלאס יש מתודות ופרופרטיז.

שיעור 58:

המשך אנגולר...
עוד דירקטיבים.
נדבר היום על איוונט ביינדינג.
בתוך תגית של HTML אפשר בכל פרופרטי לתת זוג של קי וליו.
הדיירקטיב NGCLASS עובד ככה
{ “key”:true\false}
אפשר להגדיר פרופרטיז של HTML בלקאס בקובץ TS באנגולר וזה יתפוס בכל המופעים שלו, זה נקרא פרופרטי ביינדינג ואת זה אנחנו ממשים עם סוגריים מרובעים "השם שנמצא בקלאס" = [SRC].

שיעור 59:

המשך אנגולר:
ראוטינג: יש ראוטינג מודול, הוא נוצר אוטומטית כשיוצרים אפליקציה.
הראוטינג מודול מפריד לך בין המודולים.
דומה לראוטינג בריאקט, מבחינת הROUTE
והלינקים עובדים אחרת הם A של HTML רק בלי HREF רק במקום זה יש דיירקטיב של המודול של הראוטינג שנקרא
ROUTINGLINK
יש באנגולר תגית קומפוננטה שאנחנו שמים פלייסהולדר במקום הקומפוננטות המתחלפו אותו קוראים
<ROUTER-OUTLET>
באנגולר עושים ראוטרים מקוננים
אם יש 3 קומפוננטות ולאחת מהם יש 3 קומפוננטות בתוכה אפשר לשים גם בבן תגית של ראוטר-אאוטלט

שיעור 60:

אובזרבבלז - חלק מריאקטיב פרוגרמינג - מעל מתכנות פונקציונאלי.
STREAMS
רצפים - רצף של פרומיסים, סטרים הוא לא צ'אנק למרות שזה ברים דומים.

RXJS
מימוש של רצפים בג'אווה סקריפט. - ספרייה שמוטמעת כבר

.