שיעור 1: היכרות עם עולם האינטרנט

```
<DOCTYPE html!>

<html>
<head>
<"style type="text/CSS>
<style/>


<head/>

<body>

<body/>
<html/>
```

```
<DOCTYPE html!>
<"html xmlns="http://www.w3.org/1999/xhtml>

<head>
<title>Bootstrap Template</title>
meta name="viewport" content="width=device-width, initial-scale=">>
<-- <!-- Bootstrap
link>
href="../../../bootstrap-4.3.1/bootstrap-4.3.1/dist/css/bootstrap-grid.min
<".css" rel="stylesheet
<head/>

<body>
h1>Hello, world!</h1> <!-- jQuery (necessary for Bootstrap's>
<-- (JavaScript plugins
```

```
script>
src="https://ajax.googleapis.com/ajax/libs/jquery3.2.1/jquery.min.js">
<</script
<-- Bootstrap plugins --!>
<script src="js/bootstrap.min.js"></script>
<body/>

<html/>
```

Galamouyal88@gmail.com

מפתח פולסטאק זה מעבר ללדעת בק אנד ופרונט אנד, צריך לדעת גם נטרוורק, דאטאבייס, מסדי נתונים.
מפתח פולסטאק טוב יכול להרים אפליקציה מאפס. End to end.
נלמד:
Html css, אלגוריתמים, JS, ברמת פרונטאנד, jsadvanced, jquery, angular, react, nodejs, mysql, mongodb, php.

עולם האינטרנט - מושגים:

Http - hyper text transfer protocol
הקליינט מבצע בקשה מאיזשהו מקור והוא מחכה לתגובה, רספונס בפרוטוקול HTTP - בדרך כלל בצד השני יש web server
ווב סרסר זה תוכנה שמותקנת על המחשב שמאזין לבקשה - בסופו של דבר השרת מאזין לנו
תוכנות ווב סרבר - Apache, iis, nginx.
הקליינט צריך דפדפן או כלי אחר כדי לבצע בקשה מהסרבר.
מטא דאטא - מידע על המידע - אם רוצים לדעת מה ההגדרות של המידע זה המאטא דאטא
המטא - מתודה של הHTTP היא get request
בדרך כלל כשאנחנו רוצים להיכנס לאתר אנחנו עושים גט-ריקווסט
השרת עונה לקליינט ב - ריספונס

השרת מדבר עם ה FILE SYSTEM - ומביא את הקובץ HTML המבוקש לקליינט (דוגמא), שרת הוא גם מחשב.
כל מה שאנחנו מתקינים על המחשב זה תוכנה.
תוכן דינמי (DYNAMIC CONTENT) - משהו שהשרת מחשב - משהו שמשתנה
סקריפט - קוד שיתקבל ויהפוך לדינמי בצד הלקוח
מושגים TCP - TCP\UDP - פיר טו פיר - השרת רק מכוון את התקשורת, הקליינטים עובדים אחד עם השני.
כתובת IP - לכל אחד באינטרנט יש כתובת IP מהספק
כתובת URI - UNIFORM RESOURCE IDENTIFIER - הסכמה
scheme://domain:port/path?query_string

דומיין עולה כסף
הפורט הוא בדיפולט 80 - הרבה פעמים לא רואים את הפורט - הפורט תמיד הוא יצוג מספרי
HTTPS - HTTP SECURE
חיבור מאובטח - מוצפן - קשה יותר לדוג את ההודעה שהקליינט שולח לשרת דרך מפתח (צופן) שיש ללקוח ולווב סרבר

ה PATH - מביא אותך למקום ספציפי בקוד
עוד פרוטוקול FTP - כ FILE TRANSFER PROTOCOL - הקליינט מדבר ישר עם המערכת קבצים של השרת.

שני פורמטים להעביר את המידע: XML ו JSON
י XML - EXTESSIBLE MARKUP LANGUAGE - שפת תגיות - פורמט שאתה יכול לבקש בקשה של אינפורמציה ספציפית.

אנחנו בעולם הווב והJS נעבוד הרבה יותר עם JSON.

<u>שיעור 2: צד לקוח HTML</u>

משתמש קצה - לקוח - צד לקוח - אנחנו בונים את מה שצד לקוח רואה, משתמשים לתכנת את זה בHTML - שפת תגיות שפיתח הצי האמריקאי.

פ CLIENT DEVELOPMENT מורכב מ3 דברים: HTML, CSS, JS

בHTML - איך יראה הדף - עיצוב, מראה

כשנרצה להמשיך לעצב את הHTML - לשנות צבעים, רספונסיבי, לבנות סט חוקי עיצוב נשתמש ב CSS

ב JS - פיתוח, כתיבת קוד - שפה מאוד מבוקשת - הופך את ה HTML וה CSS מסטטי לדינמי.

בסופו של דבר HTML הוא קובץ - סיומת הוא .HTML או .HTM - הדפדפנים יודעים לקרוא קבצי HTML

כשפותחים קובץ מהמחשב שלך הוא לא נותן כתובת IP אלא כתובת אבסולוטית בתוך המחשב שלך.

ללמוד את הפקודות של ה CMD

שבלונת HTML

```
<DOCTYPE html!>
<html>
<head>

<html/>
<body>

<body/>
<html/>
```

הפקודת הפעלה ב CMD של הוויז'ואל סטודיו קוד הוא CODE .
כשמסיימים פקודת דוקטיים HTML הכוונה ל5 HTML.
על כל תגית מכולה שנפתחת ב HTML צריך גם לסגור את התווית.
תגית HEAD הוא מידע על הדף - מטאדאטא על הדף - כמו מה שכתוב בלשונית בדפדפן

תגית מכולה - יכולה להכיל בתוכה גם תוכן וגם קוד.

דוקטייפ הוא לא תגית מכולה

בBODY - נראה את התוכן של הדף

כל תגית תיפתח עם משולש גדול מ ונסגר עם משולש לצד השני. תגית עם / תהייה סגירה של תגית מכולה.

מה שנמצא בין הפתיחה והסגירה של כל תגית מכולה הוא הערך של התגית

תגית <P> - פרגרף נותן לתת פסקה ארוכה

תגית <TITLE> - נותנת שם לכרטיסיה בדפדפן - תמיד יהייה בתוך ה HEAD

בתגית לא מכולה הסלאש יהייה בסוף התגית

תגית </HR> - עם הספלא בסוף הHR עושה קו באמצע הדף

את הATTRIBUTE של התגית בעצמה נכתוב בתוך התגית עם רווח רק בפתיחה של התגית. בתגית לא מכולה האטריביוט יהייה בסוף, לפני הסלאש

אטריוביוט BGCOLOR - נותן בקגראונד קולור.
לא כל התכונות חלות על כל התגיות. על כל ATR יהייה אחריו = ומיד אחריו הערך שלו למשל
<P ALIGN="CENTER">
תגית
 - ברייק - מוריד שורה
אם שמים את ה ATT ה TITTLE על <P> זה עושה TOOL TIP נחמד.

התגית <HEADER> - יש H1 H2 וכן הלאה, לכל מספר זה אומר שהוא בגודל אחר
התגית <DIV> - קונטיינר - אם אתה רוצה לשנות חלק שלם בדף אתה נותן את האטרביוט בדיב

בתוך אטריביוט כמו STYLE, אפשר לשים KEY ו VALUE בצורה KEY:VALUE
אפשר לשים ; בין הקיז השונים ואז אפשר לשים יותר מקי אחק באטריביוט למשל ,FONT-SIZE

גודל יחסי לפונט סייז שנקבע נותנים כ EM בתוך הסטייל=פונט-סייז:2EM

שיעור: לעשות קורות חיים כHTML:

H1 - cv {name{
P
Id
Tel
Address
[size 1em, center, color-blue]
H2 {place} {yearrs} 2
h5
{description} black 2em

כל דסקריפשין בצבע אחר

H3 - {army expirience}, army green 2em
H3 - {languegues}, 1-10 2em

שיעור 3:עוד HTML

יש עוד KEYבתוך האטריביוט STYLE והוא LATTER SPACING - מרחק בין כל אות

תגית A - לינק - אטריביוט HREF - מהו הלינק שאנחנו רוצים להשתמש בו - גלישה לדף HTMK אחר קורים לו ANCHOR.

אטריוביוט נוסף TARGET - האם לפתוח בעמוד חדש או באותו עמוד הערכים הם _BLANK ו_SELF

מה ההבדל בין בלוק לאינליין

אצל בלוקים, כל סיבלינג של הבלוק ירד שורה למטה. באינליין כל אח שלו יתפוס את האורך שלו והאינליין הבא ימשיך מיד אחריו

התגית IMG - נותנת להעלות תמונה, ה SRC הוא אטריביוט שבו נותנים URL או פת' רלטיבי, HIGHT ו WIDTH

בקונסולה של הF12 עם עושים אינספקט לאלמנט אפשר לכתוב את זה לראות מה זה עושה, אחר כך להעתיק את השינויים ולבצע אותם בעורך HTML שלנו.

עימוד נכון חוסך עבודה אחר כך - הכל בתוך >DIV< נכון ואטריביוטס של עיצוב עדיף תמיד לקבוע בדיב
תגית <SPAN> - תגית אינליין, לא יורדת שורה - אם רוצים לשנות משהו ספציפי בתוך פסקה או משהו כזה אז לשים אותו בספאן

תגית >B< - תגית אינליין שעושה בולד על הפונט

בן לפני אבא בענייני עיצוב - אם נתת לדיב הראשי צבע אחד אבל לבן שלו נתת צבע אחר אז הבן יקבל את הצבע שלו.
לא לתת סתם אטריביוטס לאבות, לראות שהכל קשור

שיעורי בית: ליצור דיב של 10 דמויות עם תמונה, תיאור ולינק - הלינק יוצא לדף אחר עם אותו תוכן כמו הדמות רק עם התמונה בגודל מלא ועוד פסקה ולכל דף לחזור חזרה לאינדקס - אפשר בדף הנוסף לעשות תמונה אחרת -לעשות גם טבלה של נתונים לכל דב - לשים גם בכל תמונה אטריוביוט ALT - כיתוב למקרה שהתמונה נשברת

-אטריביוט ALT בתוך IMG

שיעור 4: עוד HTML

המשך של HTML

לעשות HOVERING - אטרביוט של טייטל - כשעומדים על משהו ומופיע כיתוב, או משהו רץ.

ליסטים ב HTML - אם רוצים להראות אינפורמיה כרשימה, UL - UNORDERED LIST ו OL - ORDERED LIST.
עוד אחד יש זה LI - LIST ITEM - כל אייטם שאני ארצה להציג נשתמש ב LI.

אנאורדרד - נותנת נקודות לכל אייטם ברשימה - בוליטן
אורדרד - נותנת מספרים לכל אייטם ברשימה

אטריוביוט ב LIST_STYLE_TYPE - ORDERED - נותן אפשרויות שונות לסדר את הרשימה המסודרת - דריסה של הסטייל הדיפולטיבי שמגיע

אחוזים באטריוביוטס תמיד יהיו ביחס לאבא של האלמנט

ראשית צירים בHTML תמיד תהייה בשמאל למעלה - אופקי X, אנכי Y

יכולת לעשות דברים שקופים - OPACITY - ערכים מ 0 ועד 1 - נמצא כ KEY בתוך הסטייל

עוד KEY בתוך הסטייל הוא POSITION - ערך מוכר הוא FIXED - הוא לא מתייחס מבחינה מיקומית לאלמנטים שלפניו ואחריו.
אחד נוסף הוא RELATIVE - תמיד נמצא רלטיוי יחסית לאבא שלו
אחרון ABSOLUTE - רלטיוי לאבא הראשי

כדי ש פיקסד יעבוד - שני האלמנטים שרוצים לשים באותו מקום צריכים להיות עם הKEY של פיקסד

Z-index and Y-index

אפשר להזיז אלמנטים על המסך רלטיוית לאבות שלו - מתחיל מ0 ואפשר ללכת קדימה או אחורה

תגית סטייל - חוץ מאטריביוט, סטייל יכול להיות גם תגית - נקרא להשתמש אינטרנל סטייל של CSS
, אלמנט מכולה שנכניס לתוכו את החוקים במקום באנליין, כל בלוק נמצא בתוך { } - לפני המסולסלות יהייה על מי חל החוקים של הסטייל.
תמיד אינליין ידרוס אינטרנל ואינטרנל ידרוס אקסטרנל

טבלה - אלמנט TABLE - אלמנט מכולה - Thead - שורת כותרת של הטבלה ו TBODY - כל שורה של הטבלה כל ריבוע בטהד זה TH - טייבל הדר. כל ריבוע בטייבל בודי זה TR - טייבל ROW - בכל טייבל רוו יש TD - טייבל דאתא

בתוך התגית הראשית של הטבלה יש אטריוביוט BORDER - שלו יש ערך במספרים מ1 עד 5, עובי הגבול

מספר הTH זה מספר ה TD בכל TR

כשאנחנו מורידים מאיפשהו ריבוע אפשר לאחד שני רשומות באותה שורה עם COLSPAN - אטריביוט בתוך TD

שיעור 5: עוד HTML

אלמנטים שהוגדרו עם פוזישן פיקסט, נשארים תמיד באותו מקום במסך. רלטיוי רק למסך.
אלמנטים שהוגדרו עם פוזישן אבסולוט - רלטיוי לאבא הראשון שהוא רלטיוי - כל אבא שהוא לא סטטיק
אטריביוטים בתוך DIV שהם CLASS ו ID. אפשר לתת חוקים לCLASS כלשהו. CLASS הוא KEY והוא שווה ל VALUE. בתוך DIV אפשר להחיל חוקים של יותר מ CLASS אחד.

אלמנט ID - אטריביוט מאוד חשוב, מזהה IDENTIFIER יחודי בדף. אפשר לחלק דיבים לתפקידים יחודיים.
לא כותבים רווחים בID - עם קו תחתון למשל TABLE_CONTINER.
אם ניתן יור מפעם אחת את אותו ID האינטרנט יקרא רק את האחרון למשל

בקונסולה אם נחפש את הID שנתנו הוא יגיד לנו מה יש בתוכו

כשמגדירים CLASS בSTYLE מגדירים אותו ע"י נקודה בהתחלה לדוגמא
.CLASS
בסטייל ID תחיל עם סולמיתץ בדב טולז אפשר לראות מאיה אלמנט קיבל מה (קלאס או ID ,(ID תמיד יהייה יותר חזק מ CLASS.

אם רוצים לתת שני קלאסים לאותו אלמנט נותנים את שניהם בתוך אטריביוט קלאס עם רווח בין שניהם
class="class1 class2"
במקרהה הזה אם יש וליוז דורסים האינטרנט יקרא ויתייחס אל האחרון שאנחנו נותנים.
כשמגדירים חוק ב CSS בין שני סלקטורים ויש רווח - הCSS יחיל את החוקים על כל הבנים של אותו סלקטור והבנים שלהם לכל הדורות
לדוגמא לתת בDIV
CLASS="P-RED P"
אם במקום רווח עושים < (GT&) אז החוק יהייה תקף רק לבנים ולא לצאצאים שלהם.

אם במקום רווח עושים + אז מי שיקבל את החוק זה רק האח הראשון.

טפסים - FORMS - תגיות שמקבלים אינפוטים מהלקוח. אינפוטים יכולים להיות מקלדת, עכבר, מיקרופון.
נדבר על אינפוטים מהמקלדת.
תגיתFORM - עבודה נכונה
ברגע שמתקבל אינפוט מהלקוח האטריביוט ACTION="" אומר מה נעשה עם האינפוט. אטריביוט נוסף הוא METHOD - אפשר לעשות GET או POST
האינפוט \INPUT - הוא תגית שפותחת שדה שהלקוח ממלא. אינפוט יכול להיות טקסט, כפתור, בחירה אחרי האינפוט עולה אטרביוט TYPE.
לכל אינפוט יש TYPE - טקסט זה הדיפולטיוי. Kלכל אינפוט יש VALUE - מה יכתוב או יבחר, הVALUE הראשוני הוא פלייס הולדר
אפשר לשים גם פלייס הולדר עם הוראות
לכל אינפוט צריך להיות גם שם NAME, דרכו נזהה את האינפוט - לשליחה של האינפוט של הלקוח נתייחס לשם כפתור SUBMIT, עוד אינפוט יבצע שליחה של כל הטפסים(אינפוטים) שהוכנסו.
הPROP של ACTION מגדיר לאן המידע ישלח ומה נעשה עם המידע?
הקונבנציה של השליחה היא
[name]="[value]"&
אם לא נגדיר NAME לא נקבל את האינפורמציה של שהלקוח נתן.
אפשרויות לסבמיט - ע"י אינפוט סבמיטת אפשר ע"י אלמנט מכולה BUTTON
בטן עם טייפ באטן לא יבצע סאבמיט.
אינפוט SELECT - נותנים מכולות OPTION, לא לשכוח לתת בVALUE את השם של הבחירה.
אינפוט RADIO עם שתי מכולות OPTION עושה אותו דבר, רק שמשהו תמיד צריך להיות לחוץ - תמיד הרדיו צריכים להיות עם אותו השם. ברדיו צריך גם אטריביוט CHECKED שהVALUE שלו צריך להיוצ CHECKED
אינפוט נוסף זה CHECKBOX - נותן ערך בוליאני, או כן או לא, אם לא לוחצים על הכפתור אז אין VALUE
לאינפוט NUMBER אפשר לשים אטריביוטים MIN ו MAX
אינפוט נוסף זה RANGE, נותן מיטר

מה עושה תגית ifame

שיעור CSS:6

ניווט בתוך הדף עם # ו HREF
ברגע שבHREF הוא רואה # הוא מחפש את אותו ID בדף
חוקיות בסלקטורים שלCSS - עם : אחרי האלמנט נותנים את הסלקטור
פונקציית (N) אם רוצים שכל פעם הילד, האות או כל אלמנט יעשה את הבא אז עושים chlid(2n) ככה זה יתן לכל סקטור שני והלאה

:NTH-CHILD)5N(

לחפש ב css selectiors
פאדינג ומרג'ין - הסדר ששמים את הכיוונים top rght bottom left. אם נותנים ערך אחד הוא נותן לכולם
מרג'ין - מהאלמנט החוצה
פאדדינג - מהאלמנט פנימה.
אפשר להוריד את הקו התחתון של לינק עם TEXT DECORATION -NONE.
יש אטריוביט CSS של DISPLAY וקוראים אותו INLINE-BLOCK - איתו אפשר להזיז את הפרופורציות של אלמנטים ועדיין שלא יהיו כבלוק
בורדר רדיוס - לעגל פינות
נעדיף לא לעבוד עם תמונות IMG אא עם גרפיקות ו SVG
בבית לקרוא את הויקיפדיה של SVG באנגלית
גרפיקה וקטורית לא מתפקסלת - היתרון הגדול שלה שומרת על הרזולוציה שלה
הSVG הוא אלמנט מכולה, לא מכניסים אליו אלמנטים של HTML, מתחיל מבחינת צירים 0 0 למעלה משמאל להיכנס ל DRAWIO
הSVG תמיד יראה בשכבה העליונה את האלמנט האחרון שכתוב, אם רוצים לשחק עם האלמנטים צריך לשנות את הסדר של האלמנטים בSVG

שיעור 7:

כל מיני דברים וסוף של HTML
טרנזישן דרך CSS - דרך איוונטים למשל דרך :HOVER. צריך את הפרופ שיגדל, ה KEY שישתנה
למשל ה WIDTH, דבר נוסף שנצטרך זה את ה זמן של הטרנזישן, את זה ניתן עם TIME:, ניתן את הטיים עם S או עם MS. דבר אחרון זה את הVALUE של הקיי שאנחנו נשנה בטרנזישן.
אטריביוט בסטייל OVERFLOW - איך ננעטוף את האלמנט ואת כל הילדים שלו הקיי AUTO מנסה לעטוף את האלמנט (האם הדיב יעטוף את הילדים). קיי נוסף זה SCROLL - פותח סרגל גלילה - לנסות להימנע מזה.
הדיפולטזה HIDEN - הילדים לא נחתכים - יוצאים מהאבא.
אם לוקחים אלמנט פיקסט ונותנים לו TOP 0 וגם בוטם 0, הוא נמרח על כל הדף.בVALUE TRANSITION - ניתן את כמה זמן יתבצע הטרנזישן ועל איזה VALUE נזיז
למשל 4S WIDTH. אם רוצים שיהייה טרנזישן גם בחזרה נשים את הVALUE טרנזישן גם בקלאס הראשי
התגית LABAL עושה ביינדינג של הINPUT והטקסט, עושה את הכתובת והאינפוט בלוק - שמים אותו ממשלפני ה INPUT.

FLEX

נלמד BOOTSTRAP - מבוסס על האטריביוט FLEX - מכיל הרבה מאוד יכולות לסידור הדף שלנו. בוטסטרף היא ספרייה שנותנת רספונסיביות לHTML - אפשר להפוך למשהו שיתאים גם לסמארטפון וגם למסך גדול. שייכים לטוויטר. העבודה היא הרבה יותר יחסית בבוטסטרפ.

פלקס זה CSS - קונטיינר עם חוקיות וילדים - את הפלקס מכריזים בתוך הסטייל כ DISPLAY:FLEX
כל הדיבים בפלקס ימצאו ביחד אחד ליד השני באותה שורה. אפשר להגדיר אחרת ב FLEX-DIRECTION. אותו שמים על האבא, אחרי ההגדרה של הדיסלי-פלקס אפשר לשים אותו גם כ COLUM.
יש וויו של FLEX-WRAP - איתו אפשר לדאוג שהכל יכנס לאותה שורה, אם מגדירים פלקס ראפ כ WRAP הוא לא יכניס את הכל לאותה שורה. FLEX-GROW - נותן לנו לתת יחס לאלמנטים באותה שורה, או באותו מסך - אם ניתן FLEX-GROW: 1 לכל האלמנטים כולם יהיו באותו גודל, כל פעם שנותנים יחס אחר, הכל נמדד לפי האלמנט הקטן ביותר. אם ALIGN-ITEMS - אפשר להגדיר איך כל האלמנטים בפלקס יצמדו במקרה שיש הגדרה אחת שונה, בHEIGT למשל.
אומר לאייטם לשחק רק עם עצמו ביחס לאבא שלו ולא את הבנים שלו - ALIGN-SELF:FLEX-END
אם אנחנו רוצים לשחק עם דברים בכמה פיקסלים את פלקס זה לא עבודה נכונה בשבילנו.
פלקס זה עבודה עם יחסים בין אלמנטים וקונטיינרים שמטרתה היא התאמה של העיצוב למכשירים שונים
נותן לך לשחק בכיוון של האלמנטים שלך על הצירים - JUSTIFY-CONTENt

דף CSS חיצוני - פותחים קובץ חדש כ CSS ומייבאים אותו אל דף ה HTML . טוענים את הדף עם תגית LINK כשהVALUE REL=SYTLESHEET.

BOOTSTRAP.

מבוסס על סטרפ - אפשר לקחת את התיקייה של בוטסטראפ ולהכיל אותה בפרוייקט, אם אני לא רוצה אני יכול להשתמש עם CDN - לעבוד עם לינקים לבוטסטראפ כקוד של CSS ב LINK
הגריד בבוטסטרפ - שימוש בקלאסים. בוטסטראפ חלק את העמוד ל12 חלקים (בכל שורה). אפשר להגדיר מספר חלקים להשתמש לכל מערכת הפעלה שונה. בבוטסטראפ ROW הוא FLEX.
מסכים של מכשירים שונים יוגדרו לפי רזולוציה מ XS-XL.
קלאסים של דיבים
קונטיינר מול קונטיינר-פלואיד
קלאס ROW.
כל חלק מה12 נקרא COL-גודל_מספר של קוביות

שיעור 8:

הגדלים של המסכים בבוטסטראפ עובדים לפי טווחים.
את הגדלים שנותנים לפי COL-SM-12 וכו, נותנים בקלאס, בלי פסיקים
הרעיון בבוטסטראפ הוא קודם כל למובייל. מעצבים מראש למובייל ואז עושים עיצוב למסכים אחרים
הקלאס CONTAINER - תופס את כל הדף לא כולל שוליים
הקלאג CONTAINER-FLUID כולל את כל הדף
את הOVERFLOW אפשר לתת על כל דיב, גם אם יש בתוכו תמונה
הקלאס בבוטסטראפ 5-1(-MT)
פאדדינג זה PT.
אפשר לעשות מארג'ין או פאדדינג לפי גודל מסך.
ליד ה COL אם רוצים להגדיר שוליים בקוביות כותבים OFFSET-1

מבוא לתכנות:

אלגוריתמים: סט פעולות כרונולוגי שצריך לעקוב אחריהם כדי לפתור בעיה.
משתנים - מקומות בזיכרון, לא נגדיר בדרך כלל את אותו משתנה פעמיים.
משתנה מגדירים את השם שלו ואז את הערך שלו למשתנה קוראים LEFTHAND הערך שבתוך המשתנה נקרא RIGHTHAND
סטרינג - טקסט - נגדיר עם גרשיים או גרשיים אחד
קלט - סטרינג או מספר שמגיע מהמשתמש ואנחנו מכניסים אותו למשתנה.
פלט בדרך כלל יוצג למשתמש על המסך.
כשמגדירים משתנה ב JS כותבים לפני CONST

<u>שיעור 9:</u>

המשך מבוא לתכנות:
השם INT אומר מספר שלם FLOAT - מספר עשרוני
מודולו: שארית החלוקה, אם המספר הקטן הוא המספר השמאלי המודולו תמיד יהייה המספר הקטן.
המודולו תמיד יהייה בין 0 למספר הימני פחות 1.
ההפך ממדולו הוא (INT) מתקבל רק המספר השלם של החלוקה בלי המודולו.
X= INT(X)

גם בסטרינגים, ל+יש כמה משמעויות. מספרים הוא מחבר, סטרינגים הוא משרשר, לפונקציה שיש לה יותר מפעולה אחת קוראים OVERLOAD, ל + יש אוברלורד, תלוי בפונקציה.
ביוטי בוליאני - שאלה של כן או לא, TRUE או FALSE.
בתרשים זרימה IF זה מעויין. ימין FALSE. שמאל TRUE.
את השאלה הבוליאנית נשאל עם אופרטורים.
כדי לשאול האם משהו הוא זהה בIF כותבים את זה A===B.
שונה כותבים ==!
גדול שווה כותבים =< קטן שווה =>
⇔ אומר שווה זהותית< = >
השפה JS - JAVA SCRIPT

המטרה הראשונה היא לקחת HTML ולעשות משהו דינאמי, היום עוברים לדברים מאוד גדולים
השפה היא שפת סקריפט דינאמית, הריצה היא סינכרונית (שורה קוראת אחרי שורה) אלא אם כן אמרנו למחשב לעשות משהו אחר. כדי לראות קוד רץ, מריצים את השפה בדפדפן, יש שינוי קטן בין הדפדפנים השונים, אנחנו נריץ בגוגל CHROME
איך טוענים SCRIPT:תגית SCRIPT בHEAD. כל מה שאנחנו כותבים בתוך התגית זה קוד שרץ

אנחנו נטען את הסקריפט רק בצורה חיצונית, בתוך הBODY. אפשר לטעון סקריפט גם חיצונית עם SRC עם קובץ של JS.
משתנה מגדירים כ =.VAR כרגיל משתנה מתקבל או מספר או סטרינג עם "". בסוף כל שורה נסיים עם ;
כדי לבקש מהתוכנה להדפיס משהו בקונסול נכתוב () ¨CONSOLE.LOG.
לא נשתמש הרבה בקונסול לוג.
בקונסול סטרינג יהייה כתוב בשחור ומספר יהייה בכחול. אם יהייה יותר מVAR אחד באותה שורה הוא ישנה צבעים כדי להבדיל בינהם.
כשנבקש אינפוט מהמשתמש נשתמה בפונקציה ()PROMT

פרומט יקפוץ למסך למשתמש והוא יצטרך לענות עליו.
לסדר שבו אנחנו כותבים את הטעינה של הסקריפטים יש משמעות כמו לסדר של הקוד שאנחנו כותבים
פונקציה ()PARSEINT ממירה הופכת סטרינג ל INT
לולאת ()IF
סוגריים מסולסלים אחרי שאלת IF אומרת את מה לעשות במקרה והIF הוא TRUE.
מיד אחרי הסוף ה {של ה}ELSE באותה שורה
כל ה JS הוא קייס סנסיטיב
הפונקציה ALERT מקפיצה הודעות למשתמש

<u>שיעור 10:</u>

המשך מבוא לתכנות ו JS:
תנאי בוליאני מורכב - OR AND AND - ב && הביטוי הבוליאני מורכב משני תנאים, הביטוי יהייה אמת רק כששני התנאים יהיו אמת.
ב || הביטוי הבוליאני המורכב יתקיים כשמספיק שאחד מהם יהייה אמת ואז כל הביטוי יהייה אמת.

תמיד נכתוב קבצים כשהמילה הראשונה עם אותיות קטנות והאות הראשונה של המילה השנייה גדולה. גם משתנים. בתנאי מורכב השפה קוראת את התנאי הרשון קודם, אם הוא לא נכון המחשב לא קורא בכלל את התנאי השני ולכן אם יש תנאי יותר בעייתי בשביל אופטומיזציה עדיף לשים אותו ראשון. ב ||העניין הוא הפוך, אם החלק הראשון לא קורא המחשב בהכרח יבדוק את החלק השני.
הפונקציה - toLowerCase - עוזרת לפתור את הבעיה של הקייס סנסיטיב.
האפשרות ELSE IF - נותנת עוד אפשרויות לביטוי הבוליאני

תנאי מקונן: - ELSE IF - אם אתה צריך לבדוק אם משתנה הוא אחד מ3 משתנים או משהו כזה אתה עושה IF בתוך IF. לביטוי הבוליאני הלא נכון אתה מתפצל לביטוי בוליאני נוסף.
אפשר לעשות דיטקשן ב קונסול לוג כדי לדעת איפה נפלת בתנאי מקונן.

לולאות WHILE:

קטע קוד שאנחנו הולכים לרוץ עליו מספר פעמים, האיטרציות של הלולאה, עד שהתנאי לא מתקיים יותר. לולאה מורכבת מתנאי בוליאני ובלוק של קוד. תמיד נבדוק את התנאי פעם אחת יותר ממספר האיטרציות
את הסינטאקס כותבים ()WHILE
יש חשיבות לסדר בלולאה

<u>שיעור 11:</u>

משתנה בלי הגדרה ראשונית הערך שלו הוא UNDIFIEND
כדי להוסיף למשתנה את עצמו ואז עוד משתנה או מספר נכתוב כך sum=+num
פונקציה של ערך אבסולוטי math.abs.
לולאה מקוננת: לולאת WHILE בתוך לולאת WHILE אחרת, לולאה דו מימדית, מספר האיטרציות יהייה מספר של הלולאה הראשונה כפול מספר האיטרציות של הלולאה השנייה.
איפוס של לולאה תמיד יהייה מחוץ ללולאה
הסיבוכיות זה הכמות המקסימלית שהאלגוריתם רץ.

שיעור 12 מבני נתונים ב JS:

לולאת FOR - כמו לולאת WHILE, יש עוד כמה פרמטרים בריצה של ה פור, במקום לספור COUNTER שאנחנו שומרים בלולאת פור הפרמטר מגיע מראש. לכל לולאת פופר יש תנאי ו ITARATOR. ומה קורה בכל איטרציה וכמובן גם אתחול. אם יש לולאת פור מוזרב יכול להיות שהיינו אמורים לעשות את זה ב WHILE.
לולאת פור מתחילים מאתחות for (var =0; i<in; i++) כדוגמא. נגדיר תמיד את ה VAR בתוך הלולאה כדי שמתכנתיים אחרים יבינו כמו שצריך את זה.
ההבדל בין LET ו VAR - לט הוא הואמשתנה שנכון רק בתוך הפונקציה הספציפית ור לעומת זאת הוא משתנה גלובאלי ויקלט כמשתנה גלובאלי
כל משתנה גלובאלי מועמס על WINDOW.
משתנה LET - SCOPE VARIABLE חייבים לאתחל שוב אחרי הלולאה - VAR נשאר הפעם האחרונה שהשארת אותו.
משתנה LET משמיד את עצמו אחרי השימוש בפונקציה שלו.
ברגע שיש שגיעה האלגוריתם יוצא החוצה ומפסיק לעבוד.

פונקציה ISINTIGERR - if(Number.isInteger(n))k - מוודא לך שבתוך המשתנה אכן יש מספר.
כל אפשרות לבדוק אינפוטים בעייתיים, שלא נותנים לך
כל ביטוי בוליאני שרוצים להפוך הוא ! למשל (!isNaN)
במקום לכתוב if(result ===false)g אפשר לכתוב if(result)g .
איך יוצאים מלולאה מוקדם? BREAK ו CONTINUE. ברייק יהייה בתוך לולאת איף בדר"כ. קונטיניו קופץ לבדיקה, גם בדרך כלל יהייה בתוך איף.

פונקציות:

פונקציה היא חלק מהאלגוריתם, לא חייב להריץ אותה, בJS תמיד לפני הגדרת פונקציה נכתוב FUNCTION ואחריה סוגריים עגולות ואחריהם סוגריים מסולסלות.
לפונקציה נקרא בשם שלה.
חלק מהפונקציות מחזירות לנו אינפורמציה
אם יש את הפקודה RETURN בתוך הפונקציה אז זאת פונקציה עם LEFTHAND אם נחליט לשים אותה באיזה משתנה
כל מה שנכתוב בתוך פונקציה אחרי RETURN לא יתבצע - זה כמו BREAK.
את המשתנים שנכניס לתוך הפונקציה נגדיר בתוך הסוגריים והם יהיו כ משתני LET.
כשנקרא לפונקציה את מה שאנחנו רוצים להכניס למשתנה של הפונקציה גם נכתוב בסוגריים.

שיעור 13:

המשך פונקציות: שם של פונקציה אותו דבר מו משתנים, לא מתחילים עם מספר וקייס סנסיטיב.
בג'אווה סקריפט תמיד הקוד של הפונקציה נמצא בסוגריים מסולסלילם אחרי הדקלרציה של הפונקציה.
טרמינולוגיה של המשתנים:
בתוך הפונקציה אפשר לשנות ערכים גלובליים. עם לא נותנים להם LET.
להגדיר משתנים לוקאלים אפשר גם בתוך הפונקציה - אי אפשר להשתמש בהם מחוץ לפונקציה.

עדיף לא להשתמש באותם שמות של משתנים בלוקאלי ובגלובאלי.
שימוש בשליחה של פרמטרים - בתוך הסוגריים אחרי הפונקציה אפשר לקבל פרמטרים. מה שהוגדר בסוגריים לא צריך להגדיר הוגדרו לוקאלית. זהו תכנות פונקציונאלי. אם בקריאה לפונקציה לא תתן פרמטרים אז כנראה שתקבל UNDIFIEND.
אפשר לתת לפופנרציה גם משתנים דינאמים. לרוב ניתן את זה.
העברה של פרמטרים פרימיטיביים הועברים בהעתק - פרמטרים לא פרימיטיביים לא יעברו בהעתק
שורת RETURN - מתפקדת כ BREAK - ברגע שההחזרת הפונקציה נגמרה. ריטורן ריק הוא ברייק ומחזיר מספר לא מוגדר. כל מה שיש מתחת לריטרן הוא קוד שאי אפשר לקרוא.
ללא לפטהנד הרטרן ילך לחלל.
אנחנו מפרקים את הקוד לחלקים כדי לבנות אותו ויותר קל לדבג אותו, יותר קל לעבוד איתו.
אפשר לכתוב לולאת איף ככה:

Return num1>num2 ? num1 :num2;

תמיד נגדיר משתנים למעלה.
בעתיד פונקציות יהיו בקובץ אחר.
הערך NULL מסמל מקום בזיכרון שאין בו כלום, נתמודד איתו כשהמערכת תרצה להחזיר לנו משהו שלא קיים
מבחינת בדיקה בינארית של משתנה מספרי, נאל ולא מוגדר הם כמו פולס ולכן אם נתת ערך לא נכון בתוך X אז IF(X)G ל יחזיר פולס.

If (typeof result === “number”)

עוד דרך לבדוק אם מה שיש לנו בתוך המשתנה הוא סוג הווליו שאנחנו רוצים
מהדברים הפשוטים ננסה להיפטר בבדיקות כמו שיותר מהר, בגלל זה נבדוק אם דברים הם לא מספרים וכדומה.
להשתמש בקונבנציות בקוד
דברים שאנחנו רוצים לראות טרו או פולס נכתוב IFSHEKER
בשביל שפונקציה באמת תהייה רייוזבל עדיף לבדוק שהפרמטר הוא באמת הפרמטר שאנחנו רוצים

NULL AND UNDIFIEND

אנדיפיינד - כשמתשנה הוכרז אבל לא קיבל ווליו.
נל הוא אובייקט, אפשר להכריז על משתנה שהוא נל.
כל מה שלא הוגדר הוא אנדיפיינד
ערכים שהם כמו פולס: NAN " " FALSE 0 NULL UNDEFINED - כל אחד מהם בשאלת איף יתקבל פולס
== בודק את הווליו
=== בודק גם את הטייפ, נל ואנדיפיינד הם לא אותו טייפ.
לא משתמשים ב == רק ב ===.

סוויץ' קייס - יחידה לוגית בJS שמאפשרת לקבל פרמטר ועל סמך הפרמטר להריץ קייס. בכל סוויץ' קייס יש דיפולט, אם לא יהיו דברים שלא יכנסו לשום קייס

```
switch(userVote) {

   case "for": {
       voteFor++;
       break;
   }
```

```
    case "against": {
        voteAgainst++;
        break;
    }
    default : {
        voteAbstained++;
        break;
    }
}
```

כל פעם שאנחנו מגדירים משתנה שאנחנו לא עושים לו REASSIGNED - עושים אותו CONST במקום LET הגדרת משתנה קבוע. אם תנסו להגדיר מחדש קונסט זה יתן שגיאה.
בתוך לולאה אפשר להגדיר קונסט כי זה סקופ וכל פעם הוא מוגדר מחדש.
תמיד נגדיר משתנה חדש כקונסט אלא אם אנחנו בטוחים שאנחנו הולכים לעשות משהו עם המשתנה הזה.

DATA STRUCTERED

מערכים - ARRY
מקום בזיכרון שמאפשר לשמור מידע בצורה דינאמית במשתנה אחד עם אינדקס.
מערך הוא דינאמי וגדל בהתאם למה שאנחנו מכניסים אליו - במערך נחפש ונבצע דברים ע"י אינדקס, מספר סידורי.
מערך מאונדקס תמיד מ0
מגדירים כ

Let myData = []

כשמגדירים אותו ריק אז אורך המערך הוא 0.
פונקציה PUSH נותנת לנו לדחוף דברים למערך

myData.push(x)

הפונקציה תוסיף לתא הפנוי הבא את הנתון שעשית פוש, אפשר עם , להכניס יותר מתא אחד בכל פעם.
אפר להגדיר מראש את המערך עם איברים.
מערך יכול להיות גם CONST - ועדיין להשתנות, בדרך כלל מערך יהייה LET.
ברגע שאנחנו מגדירים מערך אנחנו מקבלים תכונות של מערך.
בתוך המערך אנחנו יכולים להחזיק הרבהה סוגים של ווליוז.
לדעת את אורך המערך נכתוב myData.length - תיתן בדיוק את מספר האלמנטים (לא המספר ההסידורי).
ע"H האינדקס נגיע לכל התאים.
להגיע לאחד התאים ניתן את שם המערך [] ובפנים מספר סידורי mydata[0]d.
אינדקס נשתמש במספרים בלבד.
אלמנט האחרון יהייה myData.length - 1 .
בלולאת פור נשים כל פעם את האינדקס ב אינדקס של המערך.
אם רוצים לבדוק עם משתנה הוא לא פולסי

if(!search || !Array.isArray(myData)) return false

בודק אם אנחנו מקבלים לפונקצית חיפוש שהמערך הוא מערך ושהשם לא פולסי.

שיעור 13:

להשתמש ב TYPEOF..
להשתמש ב includes() -פונקציה שרצה במערך ואומרת לך אם יש את הוליו שנתת באחד התאים. - הפונקציה לא רצה על אובייקטים, רצה או על מספרים או על סטרינגים, זה בוליאני.
סטרינג הוא מערך של תוים
אפשר לעשות .length לכל סטרינג והוא אומר לך את אורך הסטרינג.
לברר מה זה *אחרי פונקציה ב JS.
אפשר לעשות פונקציה בוליאנית ככה

Return c.length ===1

אם זה יהייה נכון תקבל חזרה טרו אם לא תקבל חזרה פולס.
לשים ברייק פויינט ב F12, דרך סורסס ולשים סמן כחול על השורה שאתה רוצה לעצור
מערך ה TYPEOF שלו הוא לא ARRAY, הוא אובייקט.
כדי לבדוק אם מערך הוא ARRAט נעשהif(Array.isArray(data) -
כשטוענים שני דפים ל HTML, אחד עם פונקציות ושני עם שאר הקוד, תמיד נטען ל HTML את הדף של הפונקציות קודם, אם לא אנחנו לא נצליח.

לעבור על המתודות של ה ARRAY ב W3SCHOOL - זה קצת מניפולציות על המערך.
פונקציה POP)(תמיד מוציא את התא האחרון של המערך.

מספר גדול של תכונות של אותו אלמנט בונים יישות כלשהיא, בפייתון זה CLASS, - זה כולל תכונות והתנהגויות.
הרעיון הוא להביא את העולם האמיתי כמה שיותר קרוב לתוכנה. בJS קוראים לזה OBJECTS
כשמגדירים אובייקט מגדירים אותו ככה

Const myCar{}

תכונות הם פרופרטיז, משתנים של האובייקט, התנהגויות הן פונקציות.
אובייקטים ב JS משתנים בי רפרנס - זאת אומרת שאם שינית אותם בפונקציה ששלחת אותם אז גם ברמה היותר גבוהה הם משתנים, גם המקור.
תכונות באובייקט אפשר להשים ביצירה שלו וגם להוסיף אחרי ההשמה שלו.
אובייקטים מכילים זוג שמכיל KEY ו VALUE. פסיק מפריד בין תכונה לתכונה, בין קי ולוליו יש :
הדרך הכי נפוצה להיכנס לתכונות של האובייקט זה עם נקודה

myCar.LP

ברגע שמשימים קיי חדש אז הגדרת תכונה נוספת.
עוד דרך לגשת לתכונה בתוך אובייקט היא להשתמש בקיי של החיפוש [] על אובייקט.

arr[index][key]===value

כשגם קי וגם וליו הם קלטים שהגיעו מהמשתמש
הDOCUMENT שלנו הוא ה DOM, אובייקט שמחזיק את כל האובייקטים ב HTML. הדוקיומנט נכנ בתוך ה WINDOW

DOM - document object model

את השימוש ב JS נעשה בעיקר כמניפולציות על הDOM

שיעור 14:

אופרציות בשיעורי בית
פונקציות של EDIT, SEARCH, ADD, DELETE.
פונקציית SPLICE() - אנחנו נותנים שני משתנים, מאיזה אינדקס אנחנו מתחילים למחוק וכמה אלמנטים ממנו אנחנו מוחקים.
לא להשתמש ב פונקציית DELETE במערכים, משאירה תא ריק EMPTY שעדיין נמצא עם הסמן

חוזרים ל DOM
כל מה שקיים מבחינה גלובאלית מועם על הווינדו.
אם משהו לא קיים על הווינדו וא נותן שגיאה
הקונטקזס, איפה אנחנו רצים - THIS
לא צריך לכתוב את הווינדו אם הולכים למשהו שנמצא ישר מתחת לווינדו
הDOCUMENT הוא רפרנס של ה HTML.
בתוך הדוקיומנטס יש הרבה מתודות, פרמטרים ופונקציות.
כל שינוי על האובייקטים של הדוקיומנט ברפרנס ב JS מתעדכן אוטומטית בדום.
אפשר להגיע לכל מקום דרך ה JS.

בתוך הדום אפשר להשתש ב GETELEMENT -
אפשר BY ID - לפי ה ID שמגיעים אליו

כשמגיעים לאלמנט רצוי תמיד נשמור אותו ב לפטהנד לפני שעושים עליו פעולות.

בחיפוש לפי TAG TANGNAME ו NAME - נקבל תצוגה של יותר מאלמנט אחד, אפילו אם יש רק אחד במסך הוא מתקבל כ COLLECTION - שזה כמו מערך.
שינו כ INNTERTEXT ו INNERHTML ימחק את כל מה שיש בתוך המכולה שלש הID
השינויים יקרו רק כשהסקריפט נטען ב HTML
איוונט ראשון - SCRIPTLOAD - כשהסקריפט נטען.
אם באיטרציה לא יהייה אובייקט אז נקבל ארור
תמיד צריך לאבטח את הקוד שעובדים עם מערכים של אובייקטים, בדיקה שבאמת יש לנו אובייקט שנכנס לפרופרטי מתאים.
יש סיכון בעבודה בנסטד אובג'קטס.
אם נרצה להכניס תוכן HTML לתוך אלמנט HTML שאנחנו נמצאים בו יש פונקציה שעושה APPAND() - יכניס אחרי הבן האחרון של האובייקט שאתהנמצא בו.
עוד פונקציה היא CREATEELEMENT()G פונקציה שמקבלת סטרינג שמחליט איזה אלמנט ניצור.
עושים אפנד רק אחרי שעושים קרייט ומוסיפים אינרטקסט.
אפשר להגיע לאלמנט BYNAME - תלוי בקונטקסט שאנחנו נמצאים, יכול למצוא את כל הספאנים בתוך דיב מסויים
האובייקטים INNERTEXT ו INNERHTML יעבדו רק באלמנטים מכולה.

איוונטים נוספים שיש בDOM הם - לחצה על העכבר, עמידה עם העכבר, מקש ימני וכו..ONCLICK
בדרך כלל אנחנו נגדיר נחפש אלמנטים ונגדיר אותם למשתנים גלובאלית.

אם אנחנו שומרים את הנתונים על הדום, כל פעם שאנחנו נרפרש את הדף המידע ימחק
יש את פונקציית REMOVE()H - מוחקת דברים מהDOM
כשנרצה למחוק לאלמנט את האבא נעשה

this.parentElement.remove

בתוך ה ONCLICK נכתוב פונקציה שלמה ולא נקרא לה ונכתוב אותה במקום אחר.
אפשר גם בתוך ה FUNCTION(E)G בתוך ה E לעשות E.TARGET.PARENTELEMENT.REMOVE

<u>שיעור 15:</u>

המשך של כתיבה לDOM.
לאינפוטים נניכנס לVALUE לא ב INNETTEXT.
פונקציה אנונימית - סטנקציה בלי שם, נותנים את זה לאירוע, כמו ONCLICK - לא נותנים שם של הפונקציה רק משתנה אחד בתוך הסוגרים
השם של המשתנה הוא המימוש שלב עם סוגריים
לכן ששמים פונקציה אנונימית על ONCLICK אז בלחיצה הוא מריץ את הפונקציה.
שווה לעשות סוג של אובייקט ואיתו לשמור את כל הנתונים החשובים שאתה.
אם נערוך אובייקטים ב JS ולא ב DOM יהייה לנו יותר קל
נחזיק מודל מאחורי הDOM ונשנה הכל בו, אחר כל שינוי נצטרך לשנות את הUI.
מבנה הנתונים יהייה מערך של אובייקטים.
יותר נוח לעבוד ככה, זה לא נמצא בDOM, זה קצת בזבזני
נגיע למצב שאת כל השינויים נעשה על מבנה הנתונים ונעשה רק מחיקה וAPPEND אחד אל ה DOM.
נפתח את המערך בציור לDOM של כל הנתונים
דיסטרקשן
לפרק אובייקט למשתנים

Const {bame, age} = user

במקום USER.NAME נכתוב רק NAME כמשתנה והמידע שהיה בתוך האובייקט כבר יהייה נמצא בו.
אפשר להשתמש במערך נתונים לחלק קטן מהאתר, ולעשות כמה דומים מזוייפים פר פרוייקט.
המילה KEYWORD, אומר תייצר מקום נוסף של האובייקט הזה, פשוט לבנות טמפלייט של אובייקט.
זה מקום אחר שבו קיבענו שבלונה של הKEYS ושולטת בהכל, FUNCTION CONSTRACTOR.
כשכותבים פונקציות של קלאסים וקלאסים בדרך כלל נכתוב את האות הראשונה גדולה בפונקציה קונסטרקטור נכתוב

This. Name = name

וככה בכל פעם באובייקט הנ"ל יהייה קיי בשם NAME עם אות ראשונה גדולה

Const someMocie = new Movie(screem, 80, horror)

ככה ניצור אובייקט.
תמיד נצטרך NEW כשניצור אובייקט ככה.
נצטרך לאפס את המערך של הדתא הנכנס תמיד לפני כל DRAW
הפונקשן קונסטרקטור הוא קלאס CLASS.
פונקציית FINDINDEX()C כמו לולאת פור רק מוצאת משהו לפי התנאי הבוליאני

שיעור 16:

כשעובדים מול מבנה נתונים, פייק-דום, הפרוייקט יותר גמיש לשינויים ולתחזוקה.
דיסטרקשן זה להוציא וליו מתוך אובייקט אחר ולהפוך אותו למשתנה שלא נצטרך לכתוב א.ב במקום זה אפשר יהייה להשתמש בו כ ב

Const {bame, age} = user

הפונקציה INCLUDES)(נותנת כמו אינדקס אוף
אם צירוף מסויים נמצא בתוך סטרינג או מערך הוא ימצא אותו.
הדרך הכי טובה לעשות את החיפוש זה הפונקציה FILTER()F

Return data.filter(car)

יתן יותר מתשובה אחת
אותה דבר עם הפונקציה FIND))(יתן לך רק תשובה אחת.
הפונקציה ONKEYUP()D - כל פעם שילחצו כפתור הוא יעשה את החיפוש ויראה אותו.
יש גם ONKEYDOWN ו ONINPUT -און קי דאון תיד יעשה את החיפוש הקודם

EVENT

פעולה
אם שמים את זה בתוך הפונקציה ב HTML הJS ידע להשתמש לכאיוונט כהפעלה של הפונקציה
מערך ריק הוא לא פולסי, הוא עדיין נחשב טרו.

LOCAL STORAGE

אם נרצה לשמור את המידע גם כשאנחנו מרפרשים את הדף

window.localStorage

הוא אובייקט ויש לו קיי וליו
וליו חייב להיות סטרינג או מספר או ערך אחד
על מערכים נשתמש d(arr)json.stringify
יהפוך אותו לסטרינג עם הפסיקים והכל. כל דבר יומר לסטרינג, גם אובייקטים

localStorage.setItem()

תקבל קיי וליו
לוקאל וליו נשמר על הדומיין שאתה נמצא עליו וכל עוד לא מחקת את הבראוזר או את הלוקל סטורג' הוא ישאר שם, ניתן לגשת אליו מאף12.

localstorage.clear()

ימחוק הכל

localStorage.getItem()

ימשוך את הווליו כל עוד נשים רק את הקיי בפנים
המידע נשמר על הברואזר ואפשר לשים עד 10 מגה.
הכל על אותו בראוזר ואותו דומיין או אפליקציה.

JSON.parse(localStorage.getItem("object"))

יחזיר את המערך להיות מערך מסרינג.
נורא נזהרים בתוך הלוקאל סטורגג' - קל לשבור את הסטרינגים ולשגע את הלוקאלסטורג'.

פונקציה

SETTIMEOUT)(SETINTERVAL)(
מתארת פעולות א-סינכרוניות בJS - הערכים הם במילישניות כשכותבים 1000 זה אומר שנייה

שיעור 17:

עד עכשיו השתמשנו במתודות לבצע דברים על ה DOCUMENT. יש אפשרות לפעול גם דרך
QUAERYSELECTOR - סלקטור הוא טייפ ב CSS כמו קלאס אידי וכדומה...
יש גם קווארי סלקטור אול - שבוחר את כל הסלקטורים הנ"ל, והגיל יביא רק את הראשון.
הסינטקס של הקווארי סלקטור הוא כמו זה של ה CSS
לID נחפש # לקלאס נחפש .
קווארי סלקטור אפשר להריץ על אלמנט ספציפי, כמו דיב או ספאן או פורם וכו'
אפשר להריץ קווארי סלקטור על קווארי סלקטור והגיע עמוק לומצוא אלמנטים ספציפים.
ADDEVENTLISTNER()H
פונקציה שאפשר להעמיס איוונטים על אלמנטים מהDOM.
הפרמטר הראשון הוא האיוונט, קוראים לזה בשם קצת אחר למשל במקום ONCLICK זה CLICK
סינטקס ככה:
span.addEventListner
פרמטר השני יהייה הפונקציה שתרוץ עם האיוונט, או עם האימפלמינטציה, או רק את השם. לא שמים בסוגריים את הפונקציה. אם נשים את הסוגריים זה יקרא לפונקציה.
פונקציה אנונימית זה פונקציה שאתה שם בתוך איוונט וכותב
FUNCTION()
אפשר לתת יותר מאיוונט אחד ויותר מפונרציה אחת בקווארי סלקטור.
אד איוונט תמיד יוסיף פונקציה לאלמנט ולא יחליף פונקציה קיימת.
TOGGLECLASS()
פונקציה שאם הקלאס המדובר מופיע הוא יוריד אותו ואם הוא א נמצא הוא יוסיף אותו.
קוסטום אטריבי ט - אדד קוסטום אטטר

טיפ: אם משהו לא עובד ולא יודעים איפה, הולכים צעד צעד ושמים קונסול לוג כדי לראות עד איפה הקוד עבד.

על FORM אפשר להשתמש בפונקציית RESET()G והוא ימחק את כל מה שצריך.

שיעור 19:

אובייקט רגולר אקספרשן REGEX - בודק שכל הטקסטים מכילים סטרינג מסויים ואוכף את זה
!!זה למשימה!! - לתאריך
להקפיץ הודעות שגיאה עם יש בעיה!!
הפתקים יופיעו בFADEIN - יתווספו משמאל לימין
צריך גם כפתור קומפליטד - COMPLETE - כשלוחצים שיהייה שינוי בUI - או שינוי בOPACITY או שיהפוך לאפור. אפשר להוסיף גם חיפוש של 3 כפתורים של ALL, COMPLETED ו OPEN.
אפשר גם בDROPDOWN.
לכל משימה אמור להיות ID - כמה שיטו - לג'נרט רנדומלי
או

“_Task” + date.now()

היום נלמד GIT

את הקוד לא נשמר אצלנו לוקאלית
גיט נותנת לנו לנהל את הקוד ולעשות גרסאות שונות
לנהל את הקוד לפי טיימליין, בעיה נוספת היא לעבוד כמה אנשים על אותו קוד

כשנקח קוד מוכן ונעבוד איתו נקח העתק שלו ולא נגע בקוד עצמו, בהמשך נתחבר אל הקוד הראשי בחזרה.
אנחנו נשמר את הסורס קוד, מה שתמיד עובד ובו אסור לגעת סורס קוד נקרא גם MASTER.
גיט אחראי על ניהול הקוד ורז'נינג שלנו.
אם התקנו GIT במחשב שלנו הוא קיים רק במחשב שלנו - כל לוקאלית
בשביל לעבוד בייחד צריך שרת, סרוויס שנותן לעשות שרGITHUNB - בין האנשים שעובדים ביחד.
בגיטהאב הכל פאבליק, כולם יכולים לראות את זה אלא אם שמת את זה כפרייבט
לקרוא קוד של מישהו אחר זה קוד-ריוויו
התיקייה שבמחשב שלנו תהייה תיקיירה "מיוחדת"
לוודא שעוא עובד

Git --version

לTASK חדש נקרא

Add-product_page

וזה ענף שיוצא מהפרוייקט

בשביל להעלות גרסאות לסרוויס צריך להשתמש כפוש

Git branch

יגיד לך על איזה בראנץ' אתה נמצא

- אומרת לך שבוצעו שינויים בקוד
- Git status
- מראהה לך את הסטטוס של התיקייה הנ"ל
- Git add ואחריו הקובץ אומר לך מה שונה ומעדכן אותו, עוד לפני העלאה לסרוויס
- גיט אד ונקודה מעלה את כל השינויים, אם נעשה על משהו שעשינו אד COMMIT השינויים ישמרו בסרוויס
- Git commit -m “changes for text”

בתוך הסוגריים יהייה את השינויים שעשית

Git log

מראה לך את כל השינויים ומתי הם קרו
גם בקומיט עוד לא העלנו את זה לשרת

Git push origin master

עושים פוש למאסטר בסרוויס

Git checkout -b

מייצר בראנץ' חדש, אחרי ה B ניתן שם לבראנץ'

Git branch

מראה לך את כל הבארנצ'ים ואיפה אתה

תמיד כדאי לעשות COMMIT לפני CHECKOUT

Git push origin head

ידחוף לך בלי לדעת איפה אתה נמצא, באיזה בראנץ' ידחוף ישר לאיפה שאתה נמצא

הפקודה GIT INIT
גורמת לתיקייה להפוך להיות סוג של FTP - יש לה איכויות מיוחדות

שיעור 20:

בגיט עדיף לעשות PUSH ל HEAD ולא לבראנץ' עצמו, כי יש סיכוי שאתה תדחוף בראנץ' למאסטר ותמחוק אותו, במצב הזה אין אפשרות לשחזר

השם של הבראנ.' תהייה השם של הפיצ'ר שאתה עובד עליו
לעבור לבארנצ'ים עושים CHECKOUT בלי -B.

יש דוקומנטציה ראשית של גיט בגיט.דוק.

הפקודה GIT-DIFF - מראה לך בדיוק איזה שינויים עשית ואיפה.

JS ADVENCED:

מערך ואובייקט - מערך הוא איטרבל אובג'קט - בJS נתייחס לכל המבני נתונים כאובייקטים.
אפשר לבדוק למערך בפונקציית ISARRAY() .
פונקציית קונסטרקטור - CLASS.
אומנם מערך הוא אובייקט אבל יש לו עוד שכבה, שהיא ARRAY, שם הוא מקבל כמה תכונות נוספות שאין לשאר האובייקטים.
האובייקט הראשי נקרא PROTOTYPE -
אפשר להוסיף דברים לפרוטוטייפ
כל CLASS - או פונקציית קונסטרקטור היא שכבה כזת כמו ARRAY שאפשר לתת לה התנהגויות חדשות וכו.

אפשר לרוץ על לולאה גם באובייקט רגיל , לרוץ על הקיז.
עדיף להשתמש בקי וליו - אז החיפוש לא רץ על מערך ומגיע הרבהה יותר מהר לאובייקט המבוקש.
חיפוש בתוך מאגר נתונים בתוך אובייקט נעשה כמו במערך עם סוגריים מרובעות

users["amirDebi"]

לקרוא על OBJECT KEYS ו OBJECT ENTRIES

אם אתה רוצה שרשר בדרך אחרת אפשר לעשות 'note_${RAND}d'

במאגר נתונים של אובייקט אתה יכול למחוק קיי עם DELETE

Delete users["amirDebi"]

הפקודה

object.keys(“users”)
תכניס לך את כל האובייקטים למערך ועליו אתה יכול לעשות LENGTH ולראות כמה אלמנטים יש.

CLASS:

קלאס הוא טמפלייט ליצירת אובייקטים
יוצרים קלאס ככה
CLASS MONSTER
לכל קלאס צריך להיות קונסטרקטור.

נפתח דף חדש של JS שנקרא לו MODELS
כאר קוראים במילת המתח NEW זה הקונסטרקטור של המחלקה.
בניגוד לפנקשן קונסטרקטור כשיוצרים מחלקה עם קלאס אפשר להוסיף מתודות למחלקה, התנהגויות

מתודה של מערך, אפשר לרוץ רק על מערך ולא על אובייקטים - FOR.EACH

שיעור 21:

מדברים עוד קצת על גיט
מה שעושים בדרך כלל ששולחים מישהו הזמנה לREVIEW עושים קומיט לאותו בראנץ' ומבקשים שיבדקו מחדש..

REGEX:
מאפשר לייצר תבנית של סטרינג מסויים.
שני דרכים לייצר רג'קס - או NEW או להשתמש ב /
Regex (rule)
משתמשים במתודת TEST()J< של הרג'קס כדי לבדוק אם משהו נכנס לרג'קס
יש כלים באינטרנט למצוא רגולר אקספרשנז
ברגע שאנחנו הופכים את הFORM למשתנה הוא הופך להיות אלמנט שבו יש קיז שהם הNAME של האינפוטים השונים.
אפשר לתת איוונט בליסטנר INPUT - כל שינוי של אינפוט יבצע את הפונקציה

שיעור 22:

גיטהאב פייג'ז - להפעיל את הפרוייקט בWEB דרך גיטהאב, הוסט של גיטהאב לדברים סטטיים.
הארכה של שבוע לפרוייקט.
אפשר לעשות גם קווארי סלקטור עם NAME של FROM ככה
querySelector(“input[name]”)

JQUERY

ספרייה הכי משומשת בעולם, פחות עובדים איתה היום, יש פתרונות היום בתוך הפריימוורק שעושים במקום.
ספריה שעוזרת לנו - המוטו הוא לכתוב פחות ולעשות יותר.
יש סיכון בזה, אבל זה ספרייה שאפשר לעשות איתה דברים מאוד יפים ב JS.
ההסינטאקס של JQUERY מתבסס על הסימן $. - זה בעצם פנקשן קונסטרקטור של ג'קווארי והוא מייצר משהו.
הסינטקס הוא:

$(selector) ----> זה בעצם כנראה ID

ההבדן סין זה לבין GETELEMEBT או QUERYSELECTOR - זה שהם מלביש על האלמנט בדולר דברים נוספים של ג'יקווארי.
יש קונבנציות שונות לעבודה בג'יקווארי מאשר לCSS ו HTML.
צריך לטעון את הספרייה כשעובדים איתה.

נדבר אחר כך - JQUERY CDN

בדיקה לג'יקווארי הוא לקחת עם $ ID של אלמנט ולשחק איתו ולבדוק שהוא באמת מקבלנשתמש בג'קווארי בפונקציה שאומרת שרק אחרי שכל האתר נטען נתחיל להריץ את הג'יקווארי.

שיעור 23:

.get()

מעביר מאובייקט של ג'קווארי לאלמנט של JS.
אם עושים APPEND לאלמנט JS שכבר נמצא בדום למקום אחר בדום הוא עובד למקום החדש ונעלם במקום הישן.

שיטה נוספת של מניפולציה על הדום כדי לשפר קצת את הביצועים היא להעביר את האלמנט ל דיספלי נון לעשות את כל השינויים ורק אז להחזיר את הדיספלי הנכון.
מתודת EACH נעשה כשמשמולו המערך או האובייקט, בתוך הפוקציה יהייה גם האינדקס וגם האלמנט.
בג'קווארי הרבה יותר מתעסקים עם הדום - אם רוצים לקבל את האלמנט JS נעשה

obj.get()

המתודות EACH תעבוד רק בג'יקווארי ו FOREACH רק ב JS ונילה
לא יעבדו אחד על אלמנט של השני.
הספרייה JQUERYUI, נותנת לך להפוך אלמנטים לDRAGABLE ועוד יכולות אחרות
צריך לדעת להפסיק איוונט ליסטינר

אפשר להחזיק את האלמנט עצמו בקונסטרקטור ככה שאפשר יהייה להגיע אל האלמנט מתוך האלמנט.
אפשר לעשות דיסטרקשן בתוך שורת המשתנים של הפונקציה.
הפונקציה GETBOUNDINGCLIENTRECT()G מביאה לך את המיקום של כל האלמנט, למצוא התנגשות עושים עם.
המרה של אלמט ג'קווארי לאלמנט JS עם .GRT נותן לך מערך
צריך להשתמש ב a.get().[0]c

PREVENT DEAFULT

מונע מהמשך האיוונט יקרה
למשל בדראג הוא ימנע מהאלמנט להמשיך להיות דבוק אל העכברמ

שיעור 24:

סטנדרטיזציה חדשה ל JS:
השפה כל הזמן מתעדכנת,

OBJECT.KEYS(OBJ)

נותן רף מערך של כל הקיז באובייקט

OBJECT.ENTRIES

מערך של צמדים של קי וליו

JS סטנדרטיזציה שיצאה חדשה של - ES6

כל שנה נותנים חידושים בשפה.
יש דפדפנים שלא תומכים ב CONST - אפשר להמיר ES6 ל ES5 ואז הקוד ירוץ גם בדפדפן שלא תומך 6.
חשוב להתעדכן כל הזמן בשינויים בקוד יש גוף שאחראי על הסטנדרטיזציות החדשות
ממיין מידע במערך Arr.filter
ES6 גם חלק מ Arr.sort
אפשר להעמיס על פרמטרים פונקציות. אפשר להגדיר פונקציה על משתנה.
במקום $, אפשר להשתמש בקידומת JQUERY עם אות ראשונה גדולה.
פונקציות על פרמטרים - ככה אפשר להעמיס את הפונקציה כמשתנה ולהעביר אותה לפונקציה אחרת עושים

```
Const sayHi = function(temp){
Return “hello” + temp
}
```

פונקציה שמעבירים כפרמטר - קול בק.
בפונקציית חץ עם אין סוגריים מסולסלות זה עושה אוטמטית RETURN

```
Const getRandom = () => `id_${math.random() * 999}`
```

אם אין פרמטרים בפונקציית חץ אפשר לעשות במקום סוגריים _. אם יש רק פרמטר אחד אפשר לכתוב אותו בלי סוגריים בכלל.
ההבל בין פונקציית חץ לפונקצייה רגילה של JS - פונקציית חץ לא יכולה לרוץ לפני שהיא תוגדר - אפשר לקרוא לפונקציי חץ רק אחרי שהיא הוגדרה.
ה THIS בפונקציית חץ הוא לקסיקלי - אין NEW בפונקציית חץ - אם שמת בפונקציה רגילה פונקציה וליו ה THIS שלך יהייה האובייקט. בפונקציית עץ.
אם לא יהייה THIS בפונקציה רגילה בתוך אובייקט הוא יחפש את את המשתנים בגלובאלי.
לפונקציית חץ הTHIS שלו הוא הווינדו

אפשר להשים פונקציות שבתוך אובייקטים להעביר אותם לסקופ גלובאלי על משתנה אחר.
ואפשר להשתמש בפונקציית H)(BIND שמשייך פונקצייה אל קונטקסט מסויים.
קשה לדבג פונקציית חץ.
פונקציית H()MAP - מייצרת מערך חדש על סמך פרמטרים מבוקשים. הרטרן של פונקציית מפ יוצר מערך חדש שאתה נותן בלפט הנד

```
Const newCountries = countries.map(){exc}
```

מפ מזקק לך מידע, לא עושים התניות בתוך מפ - תמיד תקבל מידע על כל אנטרי, איטרציה
הפונקציית פילטר - מחזיר מערך יותר קטן נכון להתניות שאתה נותן

בפילטר אתה מחזיר תנאי בסינטאקס,
מתודה נוספת היא FIND()D עושה חיפוש איטרטיבי ומחזירה אייטם אחד פי ההתניות שלך.
הפונקציה REDUCE

שיעור 25:

המרת מערך לאובייקט:
יוצרים אובייקט חדש ריק. עושים פוראיץ' למערך
מכניסים לאובייקט במקום הקונטרי ניים את כל האובייקט האיטרטיבי קאנטרי
פונקציית REDUCE()H. - הפונקציה הכי חזקה מבין כל החברה, הפונקציה של הREDUCE חוץ מהאלמנט
מקבלת עוד משתנה שיכול לעשות לך עוד פעולה אגרגטיבית במקום להשתמש ב SIDE_AFFECT
חייבים להגדיר גם את הערך ההתחלתי של VALUE. הוליו יופיע בסוגריים לפני האלמנט.
בכל איטרציה צריך להחזיר את ה וליו
המונח המרצועי של הוליו הוא ecumilator
עדיף לא לעשות התניות בתוך הפונקציה רידוס

```
Const result = countries.reduce((value, country) =>{
Return value + country.population
}, 0)
```

אפשר עם הרידוס להפוך מערך לאובייקט בקלות
פונקציית SOME - אתה שואל אם משהו נמצא במערך ומחזיר תשובה בוליאנית

העתקה עם ... - אפשר להעתיק את התכולה של אובייקט מבלי שזה יעבור עם רפרנס שלוש נקודות עם ספרד אופרטור - אם מעתיקים עם .. פרופרטיז פרימיטיבים אפשר לשנות אחד מהם מבלי שזה ישפיע על מי שהועתק אם מדובר באובייקטים אז זה כן ישפיע - נקרא העתקה רדודה עם מדובר באובייקטים כבנים, אם מדובר בערכים פרימיטיבים מדובר בהעתקה מלאה.
אם רוצים אחרי ההעתקה הרדודה להוסיף בתוך פונקציית רידוס עושים , ואחריו את הקיי והוליו שרוצים להוסיף בתוך הרידוס הקיי צריך להיות ב [].

אג'אקס:

עד עכשיו עבדנו הרבה עם הקליינט סייד
צריך להתחיל לחשוב על הקשר שלנו עם השרת
עכשיו את הדאתא שלנו נביא מהשרת עם HTTP ריקווסט..
השרת יחזיר לנו ריספונס.
נבצע ריקווסטים מתחת לשולחן, לא ברמת ה HREF.. אלא דרך הג'וואה סקריפט
הבקשות מהJS יצאו עם XHR או AJAX
בצד השרת נדבר על טכנולוגיית REST

שיעור 26:

הקונספט קולבק והרצה א-סינכרונית

דברים כמו סט-טיימאווט מסיימים להריץ את כל הקוד ואז חוזרים אל הטיים-אווט לא משנה איפה הוא נמצא בקוד.. האיוונט לוף חוזר אחורה בקוד ומסתכל בסטאק מי נכנס לשם.. אם הרצת שאר הקוד לוקח יותר משלוש שניות של הטיים אווט רק אז הקוד יחזור לטיים-אווט אך לא יחכה ויריץ ישר
אם שמים על הטיים אווט 0 שניות הוא גם ירוץ אחרי סוף הקוד.
דוגמא טובה היא אם אתה תלוי באיזר אחר שלוקח לו לעות 5 שניות אתה תחכה כדי לא להעלות תוכן לא מוכן..
כל הקולבקים יכנסו לסטאק וירוצו רק בסוף הסקריפט.

לא עושים לפטהנד לפונקציה א-סינכרונית.
איך לבקש בקשה מהשרת ?
את הבקשה מהשרת נוציא מהJS לאיזה RESTAPI ומקבלים רספונס, הבקשה היא over HTTP
הטכנולוגיה של הבקשות היא דרך אג'קס
מתודות עיקריות של אג'קס: GET POST
בפוסט נעביר JSON אג'קס אובג'קט:

```
$.ajax({
Url:”reasorce”,
Method: “GET”,
Success: function (reasult, statustext, response){
...
})
```

תשובה של 200 זה אומר שהתשובה תקינה ואפר להשתמש בדאטא, בסקסס נכתוב פונקציה כדי לבצע משהו שם:

אפשר להוסיף פונקציה של ERROR

<u>שיעור27:</u>

אג'קס מבוסס על XHR
אנחנו נבצע בקשות א-סינכרוניות ועם קולבקס נשתמש בתשובה.
לשיעור הבא ללמוד עוד פעם את כל הסטטוסים של תשובות שרת ב HTTP.
אפשר להכין מודל דום ואת הDATA לשמור שם…

CALLBACKHELL

פונקציות בתוך פונקציות שבמגיעות קריאה בתוך קריאה, כתיבה מכוערת.
יש אלטרנטיבה.
כמריצים ריקווסטים בקולבק הל הוא לא מחכה עד שתחזור תשובה לפני שהוא מריץ את הבקשה הבאה הוא מריץ תו"כ
בעיה אחת היא שאנחנו לא יכולים לדעת מתי כל הקריאות חזרו,

PROMISE:

פרומיס זה הבטחה, קודם כל זה אובייקט, מתאר תהליך א-סינכרוני, קומפיליישן או ריג'קשן של תהליך א-סינכרוני.
כל מה שעושים עם פרומיס אפשר לעשות עם קולבק, פשוט הוא נראה יותר כיי לשימוש ואפשר לשרשר אותו.
לפרומיס יש סטטוסים שהוא נמצא בהם, אם הפעולה לא הושלמה אז הסטטוס הוא בפנדינג, קומפליטד או ריז'ולבד כשהוא הסתיים וריג'קטד אז הוא לא הצליח.
שתי מתודות חשובות של פרומיס

Resolve and reject

כשאתה קורא פקודת גט באג'קס מחזירים פרומיס בשבילך
אפשר לשלוח גם ב JS ב ES6, NEW PROMISE
פונקציית THEN()L
אם משתמשים בה על פרומיס הוא יראה לך את המידע, הפונקציה RESOLVE, מגדירה את הפרומיס כרזולבד.
הTHEN יפעל רק כשהפרומיס מסתיים והמידע הגיע - בעצם זה כמו CALLBACK
עם עושים ניו פרומיס, זה פונקציה עם שתי פונקציות בפנים.
RESOLVE וREJECT.
אפשר לשרשר קטץ' וד'ן אחד אחרי השני על פרומיס, אם הקריאה ריזולבד הוא יבצע את הד'ן אם הקריאה ר'יגקט יתבצע מה שיש ב קץ'.
אפשר לשלב את האג'קס והפרומיס ביחד.

שיעור 28:

PROMISE.ALL

אם אתה שם את כל הפרומיסים שלך במערך אפשר להשתמש במתודה הזאת
היא תעבוד רק עם כל הפרומיסים יהיו ריזולבד
אם אחד מהם יפול מסיבה כלשהיא אז שום דבר לא ירוץ/
אפשר לעשות גם דיסטרקשן ברמת המערך.
לשים SERVICIES בדף JS נפרד

TYPESCRIPT - שפה של מייקרוסופט, בהיי לבל יותר, אפשר להגביל דברים
פרמטרים מסויימים חייבים להיות מסוג מסויים. ואפשר להריץ ERROR בIDI.
קבצים של טייפסקריפט מסתיימים ב .TS
הדפדפן לא קורא קבצי TS אם יש כתיבה של טייפסקריפט. לכן יש עוד טרנספורמיזציה של קבצי ה TS לJS
עדיין צריך לשים ולידציות למרות ההגבלות של הטייפסקריפט מכיוון שבזמן ריצה הטייפסקריפט לא רלוונטי
רלוונטי רק בזמן הכתיבה.
נעבוד בטייפסקריפט בריאקט

שיעור 29 :

בחלק מבקשות ה GET של אג'קס אפשר להעביר מידע לשרת עם סימן שאלה ופרמטרים

?P1=11
P2= 24
P3 = 4

BASE= או
בבקשות POST יש כחלק מהאובייקט של האג'קס קיי DATA.
שלושת הדברים החשובים של כל בקשת REST
URL METHOD DATA

המשך טייפסקריפט -
דפדפן לא יודע לפרסר טייפסקריפט, כל עוד כותבים בקובץ TS בJS הדפדפן יפרסר אותו
הטייפים הכי חשובים ב TS הם סטרינג ונאמבר
טייפ עושים עם : ואז הטייפ למשל

Const userName: string = “Gan Sela”

איך מריצים את הקוד של הTS
חבילה של ה TS-NODE.
היום יש הרבה חבילות שעושות את זה בצורה אוטומטית
לכל פרויקט TS יש קובץ CONFIG
הקובץ קוננפיג הוא קובץ JSON
קובץ TS מומר לקובץ JS ואז נטען ב HTML
יש גם WEBPACK שעושה את אותו דבר - הווובפק הוא קובץ JS
רץ בטרמינל של המחשב ולא בדפדפן NODE.JS
בנוד יש NPM - מוריד חבילות של NODE. - כמו PIP
בNPM אפשר לעשות INSTALL GLOBAL או INSTALL LOCAL
בNPM אם יש -G אז זה יהייה גלובאלי, ברי זה זה יהייה לוקאלי.
כשפותחים קובץ ג'ייסון התוכן, חייב להיות עטוף באובייקט - תמיד זה הייה אובייקט
ללמוד טייפסקריפט קונפיג: אחראי לקחת את ה TS ל JS, כל ההגדרות של התהליך
ברגע שיש קובץ קונפיג רק צריך להיות ריץ את הפקודה של הקומפיילר

TSC

וכל הקומפייל עובד לפי ההגדרות של קובץ הקונפיג
קונטרול שיפט B מקובץ הקונפיג פותח לך ליסטנר על שינויים

<u>שיעור 30:</u>

פרוייקט הבא -
הגשה גיטהאב-פייג'ז וזיפ
לא צריך XAMPP
ללמוד פרלקס סקרולינג.
2.ניווט בפרוייקט - 2ב מבוטל, לעשות 2א
לבגביל את כמות המטבעות עם SLICE לצורך הפיתוח
המחירים מול מטבעות נמצאים ב MARKET DATAו CURRENCT_PRICE
לשמור כל קריאה קודמת במשתנה.
לא להתייחס לטיימר של 2 דקות.
הגרף הוא בונוס - מה שצריך לעשות בדוחות זה לעשות דף של המידע של המטבעות

המשך טייפסקריפט:
אפשר לעשות גם לבאנדל לקוד, לאחד כמה קבצים לקובץ אחד ועושים אותו מיניפייד.
בשביל זה צריך את תיקיית WEBPACK-CLI
צריך להתקין גם את BABEL

הורשה: EXTENDS
אפשר להוריש מCLASS לקלאס אחר פרופרטיז ומתודות עם הפקודה EXTENDS
.
אין בJS הורשה מרובה.
כשמגדירים מחלקה, מגדירים את התכונות וההתנהגויות שלה.
לכל תכונות של קלאס ב TS אפשר לתת מודיפיירז
להגביל פרופרטיז מסויימים בCLASS
ע"י שימוש ב PUBLIC ו PRIVATE.
הפרייבט יכנס לתוקף מרגע שנוצר ההופעה הספציפית של הקלאס.
קיי נוסף היא SUPER והוא אם כשאנחנו יוצרים הופעה וצריך לשלוח פרטים לאבא שמוריש.
לשים בקונסטרקטור ליד משתנה בסוגריים ? הופך אותו למשתנה אופציונאלי.
בתוך הקונסטרקטור אתה צריך לתת אופציית || שאומרת במקרה שהוא לא מגיע מה לשים בתוכו.
פרמטר אופיונאלי תמיד צריך להיות בסוף.
כדי לדלג על פרופרטי אופציונאלי צריך להשים במקום הפרופרטי NULL.