**פרויקט סוף – PLANET Reviews**

**מגישים:**

**עומר לב (ת"ז 209403427)**

**מתן ראובן טל (ת"ז 208017772)**

**אריאל לניאדו (ת"ז 318930393)**

**ספיר גרסטמן (ת"ז 323116186)**

**רון קריספין (ת"ז 212131056)**

**אופיר דגן (ת"ז 208224675)**

**אלמנטים/תכונות שנלמדו עצמאית:**

**Table Colspan & Rowspan-** תכונה של טבלה המאפשרת "מיזוג" של תאים.

**&nbsp;-** non-breaking space (תו רווח).

**Image Alt-** תכונה של תמונה המציגה כיתוב במקום התמונה אם היא לא מוצגת (מסיבות שונות, כדוגמת חיבור אינטרנט איטי או קישור לא תקין).

**<p>-** פסקה. לפני ואחרי הפסקה תהיה שורה ריקה באופן אוטומטי.

**Vertical Align, Align Items & Text Align-** יישור ע"פ הציר האנכי/אופקי. Text Align מתייחס בעיקר לטקסט אבל עובד גם על אלמנטים נוספים.

**Textarea-** אלמנט טופס לקבלת טקסט ארוך מהמשתמש.

**Maxlength-** תכונה המגבילה את כמות התווים באלמנט מסוג input או textarea.

**Placeholder-** כיתוב המוצג בחלונית אלמנט input או textarea שנעלם ברגע שמתחילים להקליד על גביו ומשמש לרוב כדי להסביר למשתמש את מטרת הקלט/הנחיות שונות.

**Border Spacing-** תכונה של טבלה המגדירה את הרווחים בין התאים בטבלה.

**Display-** תכונה המגדירה את התצוגה של אלמנט מסוים (יופיע/לא יופיע ועוד..)

**Margin-** תכונה המגדירה רווחים מסביב לאלמנט.

**Transition-** מאפיין ב- CSS המאפשר לשלוט על מעברים חלקים ומעוצבים (מעין הנפשה).

**Cursor-** מאפיין ב- CSS המאפשר לשלוט על תצורת סמן העכבר בעת מעבר על אלמנט.

**Border-radius-** מאפיין ב- CSS המאפשר לעגל את פינותיו של אובייקט מרובע.

**Box-Shadow-** מאפיין ב- CSS המאפשר יצירת הצללה לאובייקט.

**<hr>-** תג HTML המשמש כחוצץ בין נושאים- בדיפולט נוצר קו הפרדה אופקי.

**Number()-** פונקציה ב- JS הממירה ערך למספר. אם הערך לא ניתן להמרה- יוחזר NaN. עבור ערכים בוליאניים- יוחזר 0 או 1 בהתאם.

**(JavaScript Object Notation) JSON-** פורמט טקסט המשמש לאחסון והעברה של מידע כמו מבנה המוכר לנו משפות אחרות (Structure). הוא נוח לשימוש מכיוון שהוא דומה במבנה הכתיבה ליצירת אובייקטים ב- JS ולכן JS יכולה בקלות להמיר את המידע לאובייקטים.

**JSON.parse()-** פונקציה הקיימת בJavaScript - הממירה מחרוזות של JSON לאובייקט.

**JSON.stringfy()-** פונקציה הקיימת ב JavaScript -הממירה אובייקט למחרוזת בJSON -.

**ForEach()-** מתודה ב-JavaScript שמפעילה פונקציה עבור כל איבר במערך. יש לציין שלא מתבצעת עבור איברים ריקים במערך.

**Document.addEventListener(event, function, Capture)-** מתודה ב- JavaScript המשמשת לצירוף "מנהל אירוע" למסמך, כלומר מה לבצע באירוע זה. הארגומנט event משמש כשם האירוע- לדוגמה click. הארגומנט function משמש כפונקציה שיש לבצע כאשר האירוע מתרחש. הארגומנט Capture אופציונלי ולא השתמשנו בו.

**Console.log()-** מתודה בJavaScript - שמדפיסה הודעה ב- console.

**Location.reload()-** מתודה הטוענת מחדש את המסמך הנוכחי. הפעולה מתבצעת כמו לחיצה על כפתור הרענן בדפדפן.

**Await-** מילה שמורה ב- JavaScript שמטרתה להשהות ביצוע פעולה של פונקציה עד ש- Promise (אובייקט המציג סיום של רצף אירועים) התקיים.

**נושא נבחר- JS Switch-Case:**

ה- Switch משמש לביצוע פעולות שונות בהתאם לתנאים שונים. תבניתו מזכירה תפריט.

להלן הסינטקס ב- JavaScript (בדומה לשפות תכנות אחרות):

switch(expression) {  
  case x:  
    *// code block*    break;  
  case y:  
    *// code block*    break;  
  default:  
    // code block  
}

ערך הביטוי שבתוך הסוגריים משווה לערכי ה- case השונים. ה- case המתאים (כלומר, שערכו שווה לערך הביטוי או עומד בתנאיו)- בלוק הקוד שתחתיו ומשוייך לו יבוצע עד הגעה למילה השמורה break.

Break מוציא מבלוק ה- Switch. יש לציין שאין חובה לעשות break ב- case האחרון מכיוון שהוא יצא בכל מקרה.

במידה ולא נמצא case תואם- יבוצע ה- default.

**מדוע בחרנו ב- Switch-Case?**

במסגרת התרגיל הראשון כתבנו פונקציה המציגה כוכבים (1 עד 5) בהתאם לביקורת שהתקבלה מהמשתמש (שיטת הניקוד).

על מנת לבצע זאת- השתמשנו בחמישה תנאי if-else שונים.

ה- Switch-Case אפשר לנו להמיר אותם ל- cases ובכך ליצור פונקציה ברורה, אלגנטית ויעילה יותר.

**הערה לגבי הקוד הספציפי שלנו: switch(true) משמעותו שההשוואות יבוצעו תחת ה- cases וה- case שימצא נכון, כלומר true- הוא זה שיבוצע.**

**התייחסות למשובים מהסטודיו:**

|  |  |  |
| --- | --- | --- |
| **משוב** | **האם התבצע שינוי** | **נימוק** |
| ממליץ לשפר את כפתור "כתוב ביקורת" | כן | הכפתורים הדיפולטיים הוחלפו בכפתורים מעוצבים ומונפשים. |
| ממליץ לשנות את הצבעים לבהירים יותר | לא | הצבעים הכהים נבחרו כדי לתת אווירה של קולנוע (בדומה לאתרים בתחום) וקיבלנו מחמאות רבות על כך. אולי בהמשך ניצור אופציה לבחירת Light/Dark Mode. |
| הוספת התראות למשתמש | לא | קיימת בדיקת תקינות קלט ומוצגות התראות למשתמש כנדרש. |
| לא ניתן להוסיף/להציג מספר ביקורות | כן | לאחר חיבור ל- Local Storage כנדרש בתרגיל זה ישנה אפשרות לכתוב ולהציג יותר מביקורת אחת. |
| אפשרות לבחור מבין מגוון סרטים | לא | בשלב זה האתר לא מחובר ל- Database ולכן בשביל להציג מספר סרטים עלינו ליצור דף עבור כל סרט.  כמובן שניתן לשמור מידע ב- Local Storage או אפילו במערך ב- JS אך שיטה זו פחות מקובלת ונכונה. נשקול לעשות זאת בעתיד בהתאם להתפתחות הפרוייקט. |
| הצבעים צריכים להתאים לבית קולנוע | לא | לדעתנו הצבעים נותנים אווירה חשוכה אך מושכת של בית קולנוע, יתרה מכך, שילבנו אלמנטים כתומים בהשראת רשת Planet, עליה מבוסס האתר. |
| חסרה בדיקת קלט | לא | קיימת בדיקת תקינות קלט כנדרש לכל השדות הרלוונטים בטופס. |
| ממליץ לעצב את האתר פחות ריבועי ובלוקי | לא | לדעתנו האתר מסודר בצורה ברורה ונגישה.  "בלוקים" (במקרה שלנו טבלאות) מסייעות לארגן את הדף. |
| ממליץ להוסיף תוכן | לא | בשלב זה אין בכוונתנו להוסיף מלל, המטרה היא לשמור על הדף פשוט, תמציתי וקולע. |
| ממליץ לשפר את שיטת הדירוג והצגתו | כן | לאחר חיבור ל- Local Storage, כתבנו פונקציה בעזרת Switch-Case המחשבת את הדירוג הממוצע בהתאם לביקורות המשתמשים והצגנו כוכבים בדף הראשי על פי החישוב. |

**התייחסות למשובים מהפרזנטציה:**

|  |  |  |
| --- | --- | --- |
| **משוב** | **האם התבצע שינוי** | **נימוק** |
| הקישור לאתר פלאנט קטן ולא בולט מספיק | כן | הלינק הוגדל. |
| יותר מידי מבוסס על מעבר לאתר IMDB | לא | המעבר לאתר IMDB אופציונלי לחלוטין (בלחיצה על הלוגו שלו). ניתן לראות את הדירוג כפי שמוצג באתר IMDB ללא מעבר אליו. |
| ממליץ להוסיף שפות לתרגום | כן | הוספנו אופציות לתרגום לגרמנית ואיטלקית. |
| חסר בביקורות תוכן נוסף | לא | האלמנטים לביקורת מוגבלים כדי לא להעיק על המבקר ולאפשר יותר מקום למלל חופשי. |

**תבנית עיצוב- Adapter:**

בחרנו להשתמש בתבנית עיצוב זו מכיוון שגילינו שקיימת מגבלה באתר שמונעת מנתח משמעותי של משתמשים להנות ממטרתו. האתר שלנו הוא אתר ביקורות ואנו רוצים לחשוף את הגולשים למגוון רחב של דעות כדי לגבש דעה מוצקה על הסרט שבו הם מעוניינים לצפות, לכן רצינו לגשר על פערי השפה של הגולשים ולאפשר להם לקרוא ביקורות ללא מגבלת השפה.

כדי להתגבר על מכשול זה השתמשנו ב- **Google Translate API.**

API משמש כערכה של קוד מוכן בעל פונקציות ומתודות שניתן להטמיע במערכת אחרת מבלי הצורך לממשה באופן עצמאי. השימוש ב- Google Translate API מאפשר לנו לתת לכל משתמש את האפשרות לתרגם את הביקורות הכתובות באתר לשפת האם שלו (בשלב זה אפשרנו תרגום לצרפתית בלבד).

בכדי לממש את ה-API בתוכנית שלנו היינו צריכים להשתמש **בתבנית העיצובית Adapter.** תבנית עיצובית זו משמשת כדי ליצור שיתוף פעולה בין מחלקות ע"י המרה של הממשק הנתון לממשק אשר יתאים לתוכנית.

בתוכנית שלנו השתמשנו בפונקציה translate() הפועלת כ- Adapter (מתאם) של הקוד הכתוב באתר לבין Google Translate API. הפונקציה יוצרת ממשק מתאים של ה- API התואם את התבנית של האתר ובנוסף מטפל בעיצוב נתונים והמרה פנימית.

הפונקציה translate כתובה בשפת התכנות JavaScript ומקבלת שני קלטים: text (הטקסט שאנו מעוניינים לתרגם) ו- lang (שפת היעד של הטקסט).

**-axios.post() Request** פונקציה מתוך ספרייה ב- JavaScript היוצרתHTTP Request מהדפדפן ומשמשת לשליחת הקלטים של הפונקציה לכתובת URL של ה- API שבה יתבצע התרגום. במשתנה translation נקבל את הפלט שניתן מהשימוש במתרגם.

**הערה:** הפונקציה translateArray() ממומשת בהתאם **לתבנית העיצובית Facade** שמטרתה לפשט ממשק עבור מערכת מסובכת ולהסתיר את המערכת הזו. בפונקציה זו אנו מייצרים ממשק פשוט כדי לתרגם מספר שדות ב- LocalStorage.

**תכנות מקבילי:**

ברצוננו שהביקורת שהמשתמש כותב תהיה תמציתית יחסית ובנוסף עקב מגבלות מקום ב- Local Storage / חיבור ל- DataBase בעתיד אשר גורר עלויות, הקצבנו לכל תגובה מקסימום של 1000 תווים.

עקב מגבלה זו עלינו לייצר חווית כתיבה נוחה כך שהמשתמש ידע בכל זמן נתון כמה תווים כתב כך שיוכל לתכנן את המשך תגובתו בהתאם (ובכדי למנוע אי נעימות מצידו).

**מימוש תכנות מקבילי בשפת JavaScript:**

JavaScript היא שפה חד-חוטית, כלומר, מבצעת כל קטע קוד בזמנו. ישנן מספר דרכים "לעקוף" זאת, הנפוצה ביותר היא שימוש ב- **Web Workers.**

Web Worker מאפשר להריץ קטע קוד הנפרד מהקוד הראשי ובכך מייעל ביצועים ותגובתיות.

משתמשים בו בעיקר לפעולות חישוביות מורכבות, עיבוד מידע או כל קטע קוד שעלול לקחת זמן ממושך ולפגוע ברציפות הקוד הראשי (ובכך לפגוע בחווית המשתמש).

ה- Worker הוא קטע קוד נפרד שמתקשר עם הקוד הראשי באמצעות "שליחת הודעות" בין הקוד הראשי אליו ובחזרה.

במהלך תכנות ה- Worker נתקלנו בבעיה הקשורה בהגבלות בטחון של הדפדפן. **הדפדפן מגביל גישה לקבצי קוד שמאוחסנים לוקלית (בדומה לאתרנו) כך שלא הצלחנו לייצר Worker.**

על מנת להתגבר על מכשול זה היה עלינו לתקשר עם שרת נוסף באמצעות כלים כמו Node.js, פייתון או הרחבות כאלו ואחרות שלא למדנו.

עקב מורכבות העניין וע"פ דרישת הקורס, יצרנו פונקציה **המדמה תכנות מקבילי** ללא שימוש ב- Worker.

השתמשנו ב- **addEventListener** אשר מבחין בכל שינוי שמתבצע ב- textarea שבטופס. הכוונה בשינוי היא כתיבת תווים, מחיקת תווים או העתקה של תווים והדבקתם.

בתוך פונקציה זו מחושב אורך הטקסט הנוכחי ומוצג למשתמש מתחת ל- textarea את כמות התווים שנרשמו עד כה מתוך ה- 1000 האפשריים.

בכך למעשה יש תחושה שהקוד "קשוב" בכל רגע נתון ומגיב למשתמש בזמן אמת.

**הערה:** באתרנו אלמנט נוסף של תכנות מקבילי והוא פונקציות א-סינכורניות **(async).** פונקציות אלו מתבצעות מבלי לחסום פעולות אחרות. אמנם הן לא מייצרות חוטים נוספים בפועל אך הן מאפשרות עבודה מקבילית תוך שימוש אידיאלי בחומרה.

דוגמה לפונקציות אלו באתרנו: translateArray(), translate().

**אתגרים וקשיים:**

**תכנות:**

* **תרגום הביקורות:** היה מאתגר ללמוד לבד כיצד עובדים עם API. נעזרנו בסרטונים ב- YouTube ו- ChatGPT לצד שעות ארוכות של חיפוש דוגמאות והסברים באינטרנט.

אחת התקלות המרכזיות שהיו היה הצורך לרענן את הדף בכדי לצפות בטקסט המתורגם.

השתמשנו בפונקציה המובנת Location.reload() אך הדבר יצר בעיות רבות חדשות במעין אפקט דומינו.

**2 הבעיות המרכזיות היו:**

1. איפוס משתנה currentLanguage בחזרה ל- "אנגלית" לאחר רענון הדף.

בכדי לשמור על משתנה זה מעודכן בכל עת, ללא קשר לרענון הדף שמרנו אותו בשורה ה- 0 ב- LocalStroage.

1. השרתים לא הספיקו לבצע את התרגום לפני רענון הדף. כתבנו פונקצית השהייה לפני רענון של 3 שניות (כך שגם טקסט ארוך מאוד יתורגם במלואו) תוך הצגת הודעה למשתמש על התהליך והצורך בהמתנה. הפעולה לא תתבצע ללא אישור המשתמש.

* **Web Workers:** במהלך תכנות ה- Worker נתקלנו בבעיה הקשורה בהגבלות בטחון של הדפדפן. הדפדפן מגביל גישה לקבצי קוד שמאוחסנים לוקלית (בדומה לאתרנו) כך שלא הצלחנו לייצר Worker.

על מנת להתגבר על מכשול זה היה עלינו לתקשר עם שרת נוסף באמצעות כלים כמו Node.js, פייתון או הרחבות כאלו ואחרות שלא למדנו.

עקב מורכבות העניין וע"פ דרישת הקורס, יצרנו פונקציה המדמה תכנות מקבילי ללא שימוש ב- Worker.

**עבודת צוות:**

* **יחס זמן אל מול השקעה:** נוכחנו לדעת שעל מנת להגיע לתוצאה הרצויה, במיוחד מבחינה עיצובית, נדרש זמן רב להתאמות. עולם ה- Web הוא אינסופי ונאלצנו להגביל את עצמנו מבחינת כמות האלמנטים והרעיונות שרצינו לממש באתר עקב מגבלות הזמן בסמסטר עמוס ומקוצר.
* **תיאום מפגשים בין 6 אנשי צוות:** לראשונה עבדנו בקבוצה כה גדולה. נאלצנו להתמודד עם חלוקת עבודה בין כל אנשי הצוות על אף קישוריות גבוהה בין הפונקציות והעמודים. כמו כן בשל הסמסטר העמוס, היה קושי לתאם פגישות כך שכולם יוכלו להצטרף.