בס"ד

תבניות טקסט - Regular Expressions

RegExp- זה אוביקט שיוצר תבניות טקסט.

מכריזים עליו ב js כך:

---

var rExp = /pattern/options **<u>או</u>**

var rExp = new RegExp("pattern","options")

הפרמטר options האם האובייקט יתייחס לכל הטקסט והאם הוא לא יתייחס להבדל בין אותיות קטנות וגדולות.

כדי לסמן שהאובייקט יתייחס לכל הטקסט כותבים את האות ה-g (מהמילה global).
כדי לסמן שהאובייקט לא יתייחס להבדל בין אותיות גדולות לקטנות כותבים את האות i.
ניתן להשתמש בשניהם ביחד ע"י כתיבת gi.

```
var rExp = /pattern/gi // global & case insensitive
var rExp = /pattern/g // global
var rExp = /pattern/i // case insensitive
```

הפרמטר pattern קובע את תבנית הטקסט שאליה האוביקט יתייחס.

מאפיין זה יכול להכין סימנים מיוחדים (רגולריים) אותיות, מספרים ותווים רגילים.

<u>תווים רגולריים:</u>

<u>\</u>- מסמל את התו שאחרי או Escape sequence
לדוגמא - '\\' :יסמן את הלוכסן - '\n' יסמן שורה חדשה

^ - מסמן את תחילת המחרוזת. אם אוביקט ה regExp-אופיין כרב שורתי (multiline) אז התו ^ מסמן את התו שבא אחרי ה \n-או \r .

$ - מסמן את סוף המחרוזת. אם אוביקט ה regExp-אופיין כרב שורתי (multiline) אז התו $ מסמל את התו שלפני ה \n או לפני ה \r.

* -מסמל שהתו שקדם לו יכול להופיע במחרוזת אפס פעמים או מס' פעמים ברצף. לדוגמא zo* : מסמל את האות z ואת המחרוזת zoo .

+ -מסמל שהתו שקדם לו יכול להופיע פעם אחת לפחות או יותר ברצף. לדוגמא zo+ :מסמל את המחרוזת zoo אבל לא את המחרוזת z . -מאפשר לא שנתקע כשלמן ביאשין

? - מסמל שהתו שלפניו יופיע פעם אחת או בכלל לא. לדוגמא do(es)? :מסמל את do ו does-

W - מספרים אותיות
התחלה- pattern (תבנית של טקסט)
g/ - option אבל מה של חיפוש אני רוצה
i/ - לא רגיש לאותיות קטנות וגדולות כמו שראינו)

5\15

בס"ד

{n}
n הוא מספר אי שלילי. מסמל את התו שלפניו n פעמים בדיוק! לדוגמא o{2} :לא מסמל את o במחרוזת Bob אבל מסמל את את הדאבל o במחרוזת food .

{,n}
nהוא מספר אי שלילי. מסמל את התו שלפניו לפחות n פעמים. לדוגמא o{2,} לא מסמל את ה-oבמחרוזת Bob אבל כן מסמל את כל ה o-במחרוזת"foooooooooooooood"
{,0}שווה ערך ל*
{,1}שווה ערך ל+

{n,m}
n ו m-הם מספרים אי שליליים כאשר n קטן מ -m .מסמל את התו שלפניו בין n ל -m פעמים לדוגמא: {1,3} יסמל את ה o-היחיד במחרוזת Bob או את הדאבל o במחרוזת food או את הטריפל o במחרוזת foood אבל לא את ה o-במחרוזתgoooooooooood
{0,1} שווה ערך ל– ?

. **(נקודה)**-מסמל את כל התווים חוץ מהתו הזה n\. כדי לסמל את כל התווים כולל n\ משתמשים בתבנית [\s\S]

x|y - מסמל או את x או את y לדוגמא:

(z|f)ood

מסמל או את המחרוזת zood או את המחרוזת food.

<u>סדרות</u>

[xyz] -מסמל רק את התווים הנמצאים בתוך ה[] מתוך המחרוזת. לדוגמא [abc] מסמל את ה-a ב -plain

[xyz^] - מסמל את כל התווים שלא נמצאים בתוך ה-[] מתוך מחרוזת. לדוגמא [abc^] מסמל את האות p ב-plain

[a-z] -מסמל את כל התווים שנמצאים בין התו השמאלי למקף לתו הימני במקף. לדוגמא [a-z] מסמל את כל אותיות האלפאבית באנגלית.

[a-z^] -מסמל את כל התווים שלא נמצאים בין התו השמאלי למקף לתו הימני למקף. לדוגמא [a- z^] מסמל את כל התווים שהם לא אותיות באנגלית

(x) -ברגע שמסמנים ביטוי מסוים בסוגריים הוא *נשמר* ויהיה אפשר להשתמש בו ע"י ציון $ והאינדקס שלו לפי מס' הפעמים שהשתמשת בסוגריים.

בס"ד

[\b] - מסמל את הbackspace-

\b - מסמל את התו שבקצה המילה. לדגומא \bc: מסמל במחרוזת "is black cat" את ה c-של cat אבל לא את ה c-שלblack

\B - מסמל את התו כשהתו לא בקצה של מילה. לדגומא \Bc: מסמל במחרוזת "is black cat" את ה c-של black אבל לא את ה c-של.cat

\cX - מסמל את לחצן הקונטרול+(Ctrl)לחצן אחר במקלדת

\d - מסמל כל תו שהוא ספרה.

\D - מסמל כל תו שהוא **לא** ספרה

\f - מבטא את ה- formfeed

\n - מבטא שורה חדשה

\r - מבטא את מקש האנטר(Enter)

\t - מבטא את ה-טאב(Tab Character)

\v - מבטא את הטאב האנכי(Vertical Tab Character)

\s - מבטא כל מרווח לבן.

\S - מבטא כל תו שהוא לא מרווח לבן. שווה ערך לביטוי[^\r\n\f\v\t]

\w - מבטא כל אות באנגלית, כל מספר וקו תחתון.(_ underscore)

\W - מבטא כל תו שהוא לא אות באנגלית,מספר או קו תחתון.

\n - n הוא ערך אי-שלילי.מסמל את הסוגריים n . לדוגמא:

```
/[E-H]\w*(,)(\s)\w*\1\2/
/** \1 match (,)
\2 match (\s) */
```

שיטות

שיטת test
תחביר:

```
regExpObj.test([str])
```

בס"ד

הפרמטר str הוא מסוג מחרוזת והשיטה בודקת האם המחרוזת שמועברת כפרמטר ממלאת אחרי התבנית השיטה מחזירה ערך בוליאני (true,false).
הדוגמא הבאה למשל בודקת האם במחרוזת יש אך ורק מספרים ואותיות.

```
var rExp1 = /[a-z0-9]/gi
var strTest = "nir the king"
var bAfterPattern = rExp.test(strToTest) // return true
```

השיטה compile
דורסת את הפרמטרים שהוגדרו עם הגדרת ה regExp-ושמה פרמטרים חדשים.

```
var rExp = /[a-z]/gi; // with case insensitive
rExp.compile("[a-z]",;"g"); // without case insensitive
```

<u>שיטות של האובייקט STRING שמשתמשות בRegExp-</u>

השיטהreplace
תחביר:

```
str.replace(regExp,strToReplace)
```

- strמשתנה מחרוזת
- regExp אובייקט RegExp
- strToReplace מחרוזת שתחליף את כל הביטויים שנמצאו מתאימים לתבנית של אובייקט ה-regExp

השיטהmatch
תחביר:

```
var anArray = str.match(regExp)
```

- strמחרוזת
- regExp אובייקט regExp

השיטה מחזירה מערך של כל הביטויים שנמצאו מתאימים לתבנית של אובייקט ה -regExp

השיטה search
תחביר:

בס"ד

---

nPosition = str.search(regExp)

- strמשתנה מחרוזת
- regExpאובייקטRegExp
מחזיר את המיקום של תחילת הביטוי שנמצא מתאים ל pattern-של ה regExp-אם לא נמצא התאמה מוחזר 1 -

בס"ד

תרגול- Regex Expression

1. בפרויקט search, מצאי:

א. סימן קריאה. \!

ב. מילים שמתחילות באות גדולה. ^[A-Z]\w*\s/[A-Z]

ג. מספר שיש אחריו רווח.

ד. מילים שמסתיימות באות s.

ה. שתי אותיות o רצופות בדיוק.

ו. מילה באמצע משפט באורך בין 3 ל-5.

ז. אות גדולה בתחילת משפט. ^[A-Z]

2. בפרויקט form ערכי בדיקות ולידציה כמו במייל:

א. עבור סיסמה 8 תווים, אותיות באנגלית וספרות.

ב. עבור טלפון עם מקף אחרי הקידומת.

3. שימוש בפונקציות:

Test()

א.הוסיפי לform נייד + בדיקה.

Replace()

ב.הוסיפי replace את הקידומת של ישראל(972+) למספר הטלפון.

ג.החליפי את התווים בסיסמה לכוכביות

Search()

ג.בדקי מה אורך שם המייל (בעזרת search למיקום ה-@)אם גדול מ-10 הדפיסי בהערה "מייל ארוך".

Match()

ד. הדפיסי בהערה את סכום הספרות בסיסמה.

בהצלחה!!