

מה זה JavaScript

מבוא

JavaScript, או בעברית, ג'אווה סקריפט, היא שפת תכנות קלה (כלומר לא דורשת מאכבים רבים) שימושתית בתוך קוד (ב"כ HTML) ומאפשרת להוסיפה תכונות רבות ל-HTML לא אפשרי – למשל פתיחת חלון הודעה מתוך דף HTML.

ג'אווה סקריפט היא שפת תכנות, ככלمر יש לה תחביר (או חוקים) ואם לא נעמוד בתחביר של השפה נקבל הודעה שגיאה.

כדי לראות זאת האשיות ב Firefox Tools ציריך לבחור מהתפריט .Console Error<-Tools

ג'אווה סקריפט היא שפת סקריפט, ככלמר לא צריך לכתב קוד בג'אווה סקריפט. ג'אווה סקריפט היא לא ג'אווה. ג'אווה איננה שפת סקריפט והשימושים שלה שונים מהשימושים של ג'אווה סקריפט. הזרורים היחידים המשותפים לג'אווה וג'אווה סקריפט הם אופרטורים דומים ובני בקרה (if, while, for) דומים. בשאר הדברים אלו שפות שונות!
לחלוטין!

איך משלבים ג'אווה סקריפט?

כדי לשלב ג'אווה סקריפט בתחום HTML יש להשתמש בתאגית script באופן הבא:

```
1 <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       document.write("Hello world!");
9     </script>
10   </body>
11 </html>
```

התכונה type בתאגית script אומרת שהוא סקריפט בשפת ג'אווה סקריפט. קיימות שפות סקריפט נוספות, כגון JavaScript. אפשר לשים קוד ג'אווה סקריפט גם בתחום התאגית head. הזכר הזה שימושי עבור כתיבת גוף של פונקציות.

אפשר להשתמש בתאגית script מספר פעמים בתחום דף HTML. אפשר גם להשתמש בקובץ ג'אווה סקריפט היוצני בעורף התאגית script של התאגית script.

פקודות קוד בג'אווה סקריפט

פקודות קוד (statements) בג'אווה סקריפט ובכלל בכל שפת תכנות הן הוראות לעשות משהו. פקודות הקוד מסתתריות בד"כ בטו נקודה-פסיק (;) למטרות שואת לא חוכה ואפשר לשימוש כל פקודה בשורה נפרדת. למטרות זאת מומלץ להשתמש בג'אווה סקריפט בתחום המפורץ ;.

אפשר לקובץ מספר פקודות בבלוק בתווך סוגרים מסולסלים {}.

הערות בג'אווה סкриיפט

הערות בג'אווה סкриיפט דומות להערות בשפת C++ או ג'אווה והן:

1. עברו שורה הערת את משתמשים ב //
2. עברו מספר שורות הערת את משתמשים ב /* */

משתנים בג'אווה סкриיפט

בג'אווה סкриיפט מגדירים משתנים (variables) באמצעות הפקודה var. בנויג' לחרבה שפות אחרות לא מצינימ אט סוג המשתנה (type). סוג המשתנה נקבע לפי ההשמה שנותנים למשתנה.

דוגמא:

```

1  <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       var x = 3;
9       var car = "subaru";
10      </script>
11    </body>
12  </html>

```

מספר הערות:

3. בדוגמא הנ"ל הגדרתי שני משתנים: משתנה x שהציבתי לו את הערך 3, ומשתנה car שהציבתי לו את הערך subaru.
4. המשתנה x יהיה מסוג נומרי (מספר), והמשתנה car יהיה מסוג מחזורת (string באנגלית). ערך מסווג מחזרות תמיד מוקף בגרשיים.
5. לא חובה להשתמש ב var. אפשר פשוט לכתוב $x = 3$.
6. אפשר לחת למשתנה ערך ורק אח"כ להגדיר אותו, כלומר אפשר לכתוב $x = 3$, ואח"כ לכתוב $x = var$. ההגדרה x var לא תappa את המשתנה x ולא תתן לו כל ערך אחר, כלומר ערכו ישאר 3.
7. כמובן שאפשר לשנות ערך של המשתנה. זה נעשה בעזרת אופרטורים.

אופרטורים בג'אווה סкриיפט

אופרטורים הם פעולות (בד"כ פועלות חשבוניות כמו חיבור או חיסור) שבוצעים על משתנים וקבועים.

אופרטטור הבסיסי ביותר הוא **השמה** (assignment) והוא מסומן ע"י $=$, למשל $3 = x$.

אופרטורים נוספים:

1. חיבור משמש כדי לחבר שני מספרים: $x + 2$
2. חיסור משמש כדי להזיר שני מספרים: $x - 2$
3. כפל משמש כדי לכפוף שני מספרים: $2 * x$
4. חילוק משמש כדי לחלק שני מספרים: $x / 2$

- .5. שארית משמש כדי לקבל את השארית של חלוקת שני מספרים: $x \% 0$ אם x זוגי ו 1 אם x אי-זוגי
- .6. הוספה/החסרה $[]$ משמש להוספה או למחטה או להסרה $[]$ ממשתנה: $x++$ או $x--$ או $-x$. ההבדל בין $x++$ ל $x--$ הוא כאשר משתמשים בהשמה: x מקדם את x ב 1 אחרי ההשמה ו $x++$ מקדם את x ב 1 לפני ההשמה.
- .7. $=+=$ משמש כדי להוסיף מספר למשתנה: $3 += x$ שקול ל $3 + x = x$
- .8. $-=$ משמש כדי להחסיר מספר ממשתנה
- .9. $*=$ משמש כדי לכפול משתנה במספר
- .10. $/=$ משמש כדי לחלק משתנה במספר
- .11. $%=$ משמש כדי לקבל שארית של חלוקת משתנה במספר
- .12. שרור מחוזות: אפשר לשרש שתי מחוזות בעזרת האופרטור $+$
- .13. השוואה: האופרטור $==$ משמש להשוואה בין שני ערכים וגם להשוואה סוג המשתנים
- .14. השוואת ערך וסוג: האופרטור $====$ משמש להשוואה בין שני סוגי המשתנים
- .15. שונה: האופרטור $!=$ משמש כדי לבדוק אם ערך אחד שונה מהשני
- .16. גדול: האופרטור $>$ משמש כדי לבדוק אם ערך אחד גדול מהשני
- .17. קטן: האופרטור $<$ משמש כדי לבדוק אם ערך אחד קטן מהשני
- .18. גדול או שווה לו: האופרטור \geq משמש כדי לבדוק אם ערך אחד גדול או שווה לערך שני
- .19. קטן או שווה לו: האופרטור \leq משמש כדי לבדוק אם ערך אחד קטן או שווה לערך שני
- .20. או (AND): האופרטור $\&\&$ משמש כדי לבדוק אם לפחות אחד משני תנאים מתקיים ייחודי
- .21. לא (OR): האופרטור $\|$ משמש כדי לבדוק אם לפחות אחד משני תנאים מתקיים
- .22. פועלות על ביטים: קיימות פועלות על ביטים של מספרים, אבל לשם כך יש צורך להכיר במספרים בינאריים
- .23. תנאי: האופרטור $: ?$ משמש לתנאי והוא יכול לבוא במקום הפקודה if
- .24. קיימים עוד מספר אופרטורים לשם ביצוע פועלות מיוחדות

פקודות תנאי בג'אווה סקריפט

בג'אווה סקריפטיש שתி פקודות תנאי:

```
1 if (x == 1)
2 {
3     document.write("x is equal to 1");
4 }
```

בדוגמא הזאת בזוקים אם x שווה ל 1, ואם כן אז מדפיסים הודעה:
ונאי עם $:else$

```
1 if (x == 1)
2 {
3     document.write("x is equal to 1");
4 }
5 else
6 {
7     document.write("x is not equal to 1");
8 }
```

בדוגמה זו את מדפסים הודעה אחת אם x שווה ל 1, והודעה אחרת אם x שונה מ 1.

```
1 if (x == 1)
2 {
3     document.write("x is equal to 1");
4 }
5 else if (x == 2)
6 {
7     document.write("x is equal to 2");
8 }
9 else
10 {
11     document.write("x is neither 1 nor 2");
12 }
```

תנאי עם if :else

בדוגמה זו את מדפסים הודעה אחת אם x שווה ל 1, הודעה שנייה אם x שווה ל 2, והודעה שלישית אם x שונה מ 1 ו מ 2.

```
14 switch (x)
15 {
16 case 1:
17     document.write("x is equal to 1");
18     break;
19 case 2:
20     document.write("x is equal to 2");
21     break;
22 default:
23     document.write("x is neither 1 nor 2");
24     break;
25 }
26 |
```

בדוגמה זו (בדיווק כמו בדוגמה עם if :else) את מדפסים הודעה אחת אם x שווה ל 1, הודעה שנייה אם x שווה ל 2, והודעה שלישית אם x שונה מ 1 ו מ 2. השינויים בפעולת break גורמת ליציאת מה switch. אם לא שמים break או משיכים ל case הבא בתוך ה switch. יכולו לשבוקות break. במקרה זה אפשר לשימושם של case-case-im אחד מתחת לשני וואו לבצע פעמיים ורוצים לבצע את אותה פעולה עבור מספר case-case-im. במקרה זה אפשר לשימושם של case-case-im אחד מתחת לשני וואו לבצע את הפעולה.

בג'אווה סקריפט יש מספר סוגים של לולאות (loops):

1. `for` loop. הלולאה הזאת משמשת לביצוע פעולות מסוימות מספר פעמים, על פי תנאי שתלו依 במשתנה מסוים. לדוגמה:

```

1  <html>
2      <head>
3          <title>This is a title</title>
4      </head>
5
6      <body>
7          <script type="text/javascript">
8              var i;
9              for (i = 0; i < 10; i++)
10             {
11                 document.write("<h3>This is a line of text<br/></h3>");
12             }
13         </script>
14     </body>
15 </html>
16

```

הדוגמא הזאת מדפסה 10 שורות זהות של טקסט.

בדוגמא הזאת הגדרתי משתנה בשם `i`. בתוכה הסוגרים של ה `for` הצביתי אפס כערך התחלתי של `i`, אח"כ נתתי תנאי סיום ללולאה (בוצע את הלולאה כל עוד `i` קטן מ 10), ובסוף קדמתי את `i` ב 1.

באופן כללי, המבנה של לולאת `for` הוא:

```

18     for (initial; condition; change)
19     {
20     }

```

כאשר ב `initial` מתחילה המשתנים,

ב `condition` בודקים אם תנאי מתקיים (ואם כן או נכניםים לגוף הלולאה, אם לא אז הלולאה מסתיימת),
ב `change` משנים ערכיהם של המשתנים.

שימוש לב שיתוךן של לולאה לא מתבצע אפילו פעם אחת (אם תנאי לא מתקיים בהתחלה), או תבוצע מספר אינסופי של פעמים (אם התנאי מתקיים כל הזמן). שימוש גם בשינוי הערך מתרחש מtbody בסיום כל הפעולות בגוף הלולאה.

2. `while` loop. הלולאה הזאת מוצבצת כל עוד תנאי מסוים מתקיים. לדוגמה:

```

1  <html>
2      <head>
3          <title>This is a title</title>
4      </head>
5
6      <body>
7          <script type="text/javascript">
8              var i = 0;
9              while (i < 10)
10             {
11                 document.write("<h3>This is a line of text<br/></h3>");
12                 i++;
13             }
14         </script>
15     </body>
16 </html>

```

כמו בדוגמא של `loop`, גם הדוגמא הזאת מדפסה 10 שורות של טקסט.

באופן כללי, המבנה של לולאת while הוא:

```
19  while (condition)
20  {
21  }
```

כאשר condition הוא תנאי שכל עוד הוא מתקיים, נכנסים לגוף הלולאה.

do. הלולאה הזאת דומה ללולאת while אחור: היא מתבצעת לפחות פעם אחת. לדוגמה: .3

```
1  <html>
2    <head>
3      <title>This is a title</title>
4    </head>
5
6    <body>
7      <script type="text/javascript">
8        var i = 0;
9        do
10       {
11         document.write("<h3>This is a line of text<br/></h3>");
12         i++;
13       } while (i < 10);
14     </script>
15   </body>
16 </html>
```

כמו בדוגמאות הקודמות, גם הדוגמא הזאת מדפסה 10 שורות של טקסט.
באופן כללי, המבנה של לולאת do while הוא:

```
19  do
20  {
21  } while (condition);
```

כאשר condition הוא תנאי שכל עוד והוא מתקיים ממשיכים בלולאה.

.4. הלולאה הזאת משמשת לעבור על איברים במערך, או לסרוק חכונות של אובייקט. בהמשך הדף אני אסביר מהו מערך ומהו אובייקט.
דוגמא לולאה

```
1  <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       var x;
9       var habits = new Array();
10      habits[0] = "Dancing";
11      habits[1] = "Biking";
12      habits[2] = "Swimming";
13
14      document.write("<ol>");
15      for (x in habits)
16      {
17        document.write("<li><h3>" + habits[x] + "</h3></li>");
18      }
19      document.write("</ol>");
20    </script>
21   </body>
22 </html>
```

.5

בדוגמה הזאת אני מדפיס בעורת לולאת `for` רשימה ממוספרת של חברים מתוך מערך.
באופן כללי, המבנה של לולאת `for` הוא:

```
25 | for (x in y)
26 | {
27 | }
```

כאשר `y` הוא מערך או אובייקט ו `x` הוא משתנה שעובר על האברים במערך או באובייקט.

הערות בקשר לולאות:

1. בכל לולאה אפשר לשים פקודה `break` וזה גורם ליציאה מהלולאה.
2. בכל לולאה אפשר לשים פקודה `continue` ואו נעבור לאייטציה הבאה של הלולאה.

חלון קופצים בג'אווה סкриיפט

בג'אווה סкриיפט אפשר לפתוח חלונות קופצים (popup) ממספר סוגיים:

1. חלון הודעה (Alert)

```

1 <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       alert("Good morning man!");
9     </script>
10    </body>
11  </html>

```

בדוגמא זהה יפתח חלון עם הודעה. לחיצה על OK תסגור את החלון.

2. חלון אישור (Confirm)

```

1 <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       var res = confirm("How do you do?");
9       if (res)
10         document.write("You're OK!");
11       else
12         document.write("Are you sick?");
13     </script>
14   </body>
15 </html>

```

בדוגמא זהה יפתח חלון עם שאלת, לחיצה על OK מודפס הודעה אחת, ולחיצה על Cancel מודפס הודעה שנייה.

3. חלון להכנסת נתונים (Prompt)

```

1 <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       var name = prompt("What's your name?");
9       document.write("Good day" + " " + name + "!");
10      </script>
11    </body>
12  </html>

```

בדוגמא זהה יפתח חלון עם בקשה לכנות שם. לחיצה על OK מודפס ברכה עם השם.

פונקציה זו קתען קוד שנמצא בבלוק ומיצעים פעולה מסוימת שאפשר לקרוא להם (להפעיל אותם) מספר מקומות. במקום לשכפל קוד במספר מקומות, פשוט קוראים לפונקציה.

כל פונקציה יש שם שמוּזהה אותה, ורשימת ארגומנטים. רשימת הארגומנטים יכולה להיות ריקה. פונקציה יכולה להחזיר או לא להחזיר ערך. אם הפונקציה מחזירה ערך, זה נעשה ע"י הפקודה `return`.

דוגמה לפונקציה שמחשבת ריבוע של מספר:

```
1  <html>
2      <head>
3          <title>This is a title</title>
4      </head>
5
6      <body>
7          <script type="text/javascript">
8              function square(n)
9  {
10      return n * n;
11  }
12
13      var i = prompt("Please enter a number:");
14      document.write("The square of " + i + " is " + square(i));
15      </script>
16  </body>
17 </html>
```

בדוגמא הזאת הקריאה לפונקציה נעשית בשורה 14 (בתוך ה `write`).

חשוב מאוד: הדוגמא הזאת תעבור, אבל אם תחליפו את פעולה הכפל שבתוך הפונקציה בפעולה חיבור – לא תקבלו חיבור של מספרים, אלא שרשור של מהרוות! כאן יש צורך להפוך את המשתנה `i` מהחזרה למספר ע"י שימוש בפונקציה `parseInt()`. כדי תמיד להשתמש ב `type="text/javascript"`, יכול לחשוף גם במקרה של כפל.

אירועים בג'אווה סקריפט

אירועים הם פעולות שנעשות ע"י המשתמש (הזוז העכבר, לחיצה על כפתור וכו') או פעולות שמתבצעות בזמן מסוים (למשל כאשר דף נתען במלואו).

אפשר לתפוס אירועים ולטפל בהם ע"י מטפלי אירועים (event handlers) שהם למעשה תוכנות בתוך ת苟ית HTML. אין כל חובה לטפל באירוע כלשהו וזאת הבחירה של המתכנן בלבד אם לטפל באירוע.

סוגי האירועים:

- | | |
|-----|---|
| 1. | onblur – אלמנט מאבד את הフォוקוס. |
| 2. | onchange – התוכן של שדה משתנה. |
| 3. | onclick – קлик של העכבר על האובייקט. |
| 4. | ondblclick – דאבל קлик של העכבר על אובייקט. |
| 5. | onerror – שגיאה כאשר טענים מסמך או תמונה. |
| 6. | onfocus – אלמנט מקבל את הフォוקוס. |
| 7. | onkeydown – לחיצה על מקש בלוח המקלשים. |
| 8. | onkeypress – לחיצה או החוקת מקש בלוח המקלשים. |
| 9. | onkeyup – שחרור מקש בלוח המקלשים. |
| 10. | onload – עמוד או תמונה סיימו להטען. |
| 11. | onmousedown – כפתור של העכבר נלחץ. |
| 12. | onmousemove – העכבר זו מעל אלמנט. |
| 13. | onmouseout – העכבר יצא מעלה. |
| 14. | onmouseover – העכבר נמצא מעל אלמנט. |
| 15. | onmouseup – כפתור של העכבר שוחרר. |
| 16. | onresize – גודל של חלון או של מסגרת השתנה. |
| 17. | onunload – המשתמש יצא מהעמוד. |
| 18. | onselect – נבחר טקסט בתחום אלמנט. |

```

1   <html>
2     <head>
3       <title>This is a title</title>
4     </head>
5
6     <body>
7       <script type="text/javascript">
8         function MouseOver(e)
9         {
10           document.bgColor = "Gray";
11         }
12
13         function MouseOut (e)
14         {
15           document.bgColor = "White";
16         }
17       </script>
18
19       <h1 onMouseOver = "MouseOver(event)"
20         onMouseOut = "MouseOut(event)">
21         Move the mouse over me
22       </h1>
23     </body>
24   </html>

```

דוגמא ל `onmouseout` ו `onmouseover`

הסביר: הוספתי את סמן העכבר `onMouseOut` ו `onMouseOver` ממכנות לתגית `h1`. שתי התכונות מפנהו לפונקציות ג'אווה סקריפט. `onMouseOver` מפנהו לפונקציה `MouseOver` שימושה את צבע הרקע של העמוד לאפור, `onMouseOut` מפנהו לפונקציה `MouseOut` שימושה את צבע הרקע לבן.

casper מעבירים את סמן העכבר מעל לטקסט, צבע הרקע של העמוד משתנה מלבן לאפור, casper מוציאים את סמן העכבר מהטקסט, צבע הרקע חזר ללבן.

הערות:

1. בדוגמה העברתי פרמטר אירוע (`event`) לפונקציות, אך לא השתמשתי בו מתוך הפונקציות. תמיד כדאי להעביר את הפרמטר זהה כי יתכן שתשתמשו בו בעתיד.

2. במקרים פשוטים (כמו זה) אפשר לוותר על כל התגיות `script` והתוכן שבתוכה ולכתוב את הקוד (= `document.bgColor` בתוכן התכונות (כלומר בתוכן `onMouseOver` ו `onMouseOut`). אני לא ממליץ על זה כי זה הופך את הקוד לפחות קרייא) ולכנן לא מראה את זה.

3. בغالל ש `HTML` הוא `case insensitive` (אפשר לנוחות תגיית ותוכנות גם באותיות גדולות וגם באותיות קטנות), אפשר לכתוב `onMouseOver` או `onmouseover` (או במקרה `case` שרוויים). אני בחרתי בזווית כתיבה עבור האירועים שנדרמת `.case camel`.

ג'אווה סקריפט בניגוד ל `HTML` זה `case sensitive`: אפשר לכתוב `if` אבל אי אפשר לכתוב `If` עבור משפט תנאי.

דוגמא נוספת. הפעם ל :onmousemove

```
1  <html>
2      <head>
3          <title>This is a title</title>
4      </head>
5
6      <body>
7          <script type="text/javascript">
8              function MouseMove(e)
9              {
10                  document.getElementById("xy").innerHTML =
11                      e.clientX + ", " + e.clientY;
12              }
13          </script>
14
15      <h1 onMouseMove = "MouseMove(event)">
16          Move the mouse over me
17      </h1>
18      <br/>
19      <h2>The location is: </h2>
20      <h2 id = "xy"/>
21  </body>
22 </html>
```

הערות:

1. בדוגמא הזאת השתמשתי בclientX ו-clientY של פרמטר האירוע.
2. בדוגמא הזאת השתמשתי בDOM HTML כדי למצוא את האובייקט UX ולשנות את ערכו. הסברים על DOM בהמשך.
3. onMouseMove היא פעולה יקרה, כי עבר כל תזוזה של העכבר נוראה ארוע. מצד שני, אם הייתי משתמש ב onMouseOver הייתי מקבל רק את המקום של העכבר בזמן הכניסה לאלמנט.

אובייקטים בג'אווה סקריפט

אובייקט הוא סוג מיוחד של משתנה שיש לו את המאפיינים הבאים:

1. לאובייקט יש תכונות (properties). הכוונה היא כמו משתנה שונמצא בתחום האובייקט.
2. לאובייקט יש מethods (methods). מודה היא כמו פונקציה שונמצאת בתחום האובייקט.
3. בד"כ (אבל לא תמיד) מתחלים אובייקט בעזרת האופרטור new.

דוגמאות לאובייקטים:

1. מחרוזת (String) – זה אובייקט ששומר בתוכו אוסף של תווים, בד"כ טקסט כלשהו.
התכוונה העיקרית של מחרוזות היא – אורך המחרוזת.
מחרוזות (רשימה חלקית):

- o indexOf() – מיקום של טקסט בתחום המחרוזות.
- o toLowerCase() – הפיכת כל המחרוזת ל lower case.
- o blink() – גורם למחרוזת להבהב ע"י שימוש ב HTML Wrapper.
- o fontcolor() – שינוי צבע ומחרוזת ע"י שימוש ב HTML Wrapper.

2. תאריך (Date) – אובייקט ששומר בתוכו תאריך, כולל שעה עד לדיווק של אלףיה השניה.
רשימה חלקית של מתחוזות:

- o new Date() – ייצירת אובייקט עם תאריך ושעה נוכחים.
- o getDate() – מוחיר את היום בחודש (ולא את התאריך, בניגוד למה שעלולים לחשב).
- o getTime() – מוחיר את מספר אלףיות השנה שהלפפו מאז ה-1/1/1970
- o toDateString() – הופך את החלק של התאריך למחרוזת קראיה.
- o toTimeString() – הופך את החלק של השעה למחרוזת קראיה.

3. מערך (Array) – אובייקט ששומר בתוכו אוסף של משתנים מאותו הסוג.
למשל במקרה להזיך רשימה של משתנים x_1, x_2, \dots, x_n , אפשר להגיד x ו- $x[0]$ יהיה כמו x_1 , וכן הלאה.
בצורה הזאת אפשר למדפיס את כל איברי המערך מהתחילה לסוף בעזרת לולאה.
התכוונה העיקרית של מערך היא length – מספר האברים בערך.
רשימה חלקית של מתחוזות:

- o concat() – שורשור של מערכים.
- o () – מסיר את האיבר האחרון מהערך ומוחיר את האיבר הזה.
- o reverse() – הופך את מקום כל האברים בערך.
- o sort() – מיזן של המערך.

הערה: בג'אווה סקריפט קיימים גם מערכים רבים מדויים. אני לא נקבע כאן לנושא זהה.

4. בולאי (Boolean) – אובייקט שיכול להזיך רק שני ערכים: true או false.

5. **Math** – אובייקט המשמש לפעולות מתמטיות.
רישמה חלקית של תכונות (קבועים מתמטיים):

- E – המספר א.
- PI – הערך של ח.
- SQRT2 – שורש ריבועי של 2.

רישמה חלקית של מתודות:

- abs() – מוחזירה את הערך המוחלט של מספר.
- max() – מוחזירה את המספר הגדול ביותר ביוון בקבוצת מספרים.
- random() – מוחזירה מספר אקראי בין 0 ל 1.
- sqrt() – מוחזירה שורש ריבועי של מספר.

6. **מספר (Number)** – אובייקט ששומר מספר עשרוני.
רישמה חלקית של תכונות:

- MAX_VALUE – המספר הגדול ביותר שאפשרי בגאווה סקריפט.
- MIN_VALUE – המספר הקטן ביותר שאפשרי בגאווה סקריפט.

רישמה חלקות של מתודות:

- toFixed() – קביעת מספר הספרות אחרי הנקודה העשרוני.
- toString() – המרת מספר למחרוזת.

7. **האובייקט הגלובלי (Global)** – אובייקט שיש בו תכונות ומתודות כלליות.
רישמה חלקית של תכונות:

- NaN – ערך של NaN.
- undefined – משתנה שעדין אין לו ערך.

רישמה חלקית של מתודות:

- eval() – לבצע מחרוזות כאשר היא שורת ג'אווה סקריפט.
- parseFloat() – המרת מחרוזת למספר נקודה צפה (floating point).
- parseInt() – המרת מחרוזת למספרשלם.

ביטויים רגולרים בג'אווה סקריפט

ביטוי רגולרי הוא אובייקט שמתאר תבנית (pattern) של חווים.
התבנית יכולה להיות פשוטה (תו יחיד) או מורכבת (מספר טלפוני או כתובת דואר אלקטרוני).
הגדרת ביטוי רגולרי:

```
var re = /pattern/modifiers;
```

או

```
var re = new RegExp(pattern, modifiers);
```

כאשר:

- pattern – התבנית שאותה רוצים לחפש או להחלף.

דוגמאות לתבניות:

1. [abc] – חיפוש כל תו בין התווים שבסוגרים המרובעים.

2. [^abc] – חיפוש כל תו שאינו בין התווים שבסוגרים המרובעים.

3. [red|green] – חיפוש מילים (או red או green).

4. \h – חיפוש התו h.

5. +h – חיפוש התו h פעם אחד או יותר.

6. *h – חיפוש התו h אפס פעמים או יותר.

7. ^h – התו h מופיע בתחילת הטקסט.

8. n\$ – התו h מופיע בסוף הטקסט.

- modifiers – פרמטרים מיוחדים לחיפוש.

דוגמאות:

1. i – מתעלם מאותיות גדולות/קטנות.

2. g – חיפוש גלובלי (למקרה שה התבנית מופיעה יותר מפעם אחת בטקסט)

לאובייקט ביטוי רגולרי יש מספר מתודות:

1. () – שינוי של ביטוי רגולרי.

2. () – בדיקת התאמה למחuzeות. הפונקציה מחזירה התאמה אם היא קיימת.

3. () – בזיקת התאמה למזרזות. הפונקציה מחזירה true או false.

בנוסף, ל String יש מספר מתודות עכורים ביטוי רגולרי:

1. match() – מציאת ביטויים רגולרים במזרזות.

2. replace() – החלפת התבנית בתבנית אחרת.

3. split() – פירוק של מזרזות באמצעות ביטוי רגולרי.

4. search() – חיפוש של מקום של ביטוי רגולרי במזרזות.

```

1 <html>
2   <head>
3     <title>This is a title</title>
4   </head>
5
6   <body>
7     <script type="text/javascript">
8       var str = "Fox, ox and crox";
9       var pattern = /ox/g;
10
11     document.write(str.match(pattern));
12     </script>
13   </body>
14 </html>

```

כאן נקבל מערך עם 3 מזרזות xo.

טיימינג בג'אווה סкриיפט

בג'אווה סкриיפט קיימות 2 פונקציות זמן (טיימינג):

- `setTimeout()` – פונקציה שבעורtha קבועים שבתוך פרק זמן מסוים יקרה משהו. לפונקציה יש 2 פרמטרים: הפעולה שתקרה, ותוך כמה זמן היא תקרה (במילישניות).
- `clearTimeout()` – פונקציה שבעורtha מבטלים זמן. הפרמטר לפונקציה הוא משתנה מסוג טימר שהוגדר ע"י `.setTimeout()`

דוגמא:

```

7
8
9
10
11
12
<script type="text/javascript">
function timeOut()
{
    setTimeout('alert("Hi!")', 3000);
}
</script>

```

מספר העורות:

1. השחמייתי כאן באירוע `onLoad` (שורה 6) כדי לקרוא לפונקציה שבה יש `setTimeout()`.
2. במקום לכתוב `alert` בתוך `setTimeout()` (שורה 10) יכולתי לקרוא לפונקציה שבתוכה יהיה ה `text`.
3. `setTimeout()` גורם לפעולה חד פעמי. אם רוצים שהפעולה ת חוזור על עצמה, יש לקרוא שוב ל `setTimeout()`.

אובייקטי והדפן בג'אווה סкриיפט

בג'אווה טקסטיפט יש מספר אובייקטים עבור הדפן:

1. `Window` – חוץ שפותוח וכל מוגרת שנמצאת בדף. משמש לפתיחת חלונות קופצים (`popup`), להודעות ב `status bar`, לביוזו `scroll` וכו'.
2. `Navigator` – משמש לקבלת מידע על הדפן (סוג, גרסה וכו').
3. `Screen` – משמש לקבלת מידע על המסך (רוחב, גובה, מס' צבעים).
4. `History` – משמש לקבלת ההיסטוריה ולניווט קדימה או אחורה.
5. `Location` – משמש לקבלת מידע על ה `URL` הנוכחי.

מה זה DOM?

DOM זה ראשי תיבות של Document Object Model, ומודל שמתאר את עמוד ה HTML ונוטן כלים לשינויו, ומהикаה של אובייקטים בעמוד ה HTML. קלומר בעוזרת DOM אפשר למשתמש בטלסט מוסים כאשר עוברים עם העכבר מעל הטקסט. אפשר גם לשנות את הפונט של הטקסט, לשחק עם styling ולעשות עוד המון המון בדברים בעוזרת !DOM

דוגמא למספר אובייקטי DOM בג'אווה סкриיפט:

.1 – מייצג את עמוד ה HTML. לאובייקט זה יש תכונות רבות, ביניהן:

- מערך של טפסים במסמך.

- מערך של קישורים (hyperlinks) במסמך.

○ עוגיות שיש במסמך. עוגיות (cookies) הן מידע שנשמר במחשב של המשתמש ומשמש בד"כ לשימירה שם משתמש, סיסמה והעדפות שונות של המשתמש באתר מסוימים. למשל Gmail משתמשים ב-cookies כדי לשמר את כתובת הדואר האלקטרוני.

- האתר המפנה לאתר הנוכחי (referrer).

בנוסף, יש לאובייקט זה מספר מתחזות:

- write() – משמש לכתיבה תוכן במסמך. השתמשתי במתודה הזאת בדוגמאות רבות.

- ()... – משמש כדי לקבל אלמנט או מספר אלמנטים.

.2 – מייצג אלמנט בעמוד HTML. כל Tagית היא אלמנט. דוגמא לתחכונת של אלמנט:

- dir – כיוון של אלמנט (משמאלי-ימין או מימין-משמאלי).

- innerHTML – תוכן HTML בתגית.

- lang – השפה (שפה דיבור) של האלמנט.

- id – התחכונה id של האלמנט.

.3 – מייצג עיצוב של אלמנט בעמוד HTML. דוגמא לתחכונת של Style :

- backgroundColor – צבע הרקע של האלמנט.

- color – צבע הטקסט.

- font – הגופן שבו משתמשים.

- textDecoration – קו מעל, על פנוי, או מתחת לטקסט.

- width – רוחב האלמנט.

- zIndex – איזה אלמנט יופיע מעל איזה אלמנט במקרה של אלמנטים עם מיקום חופף.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script type="text/javascript">
7
8          var x = Math.abs(-5);
9          var r = Math.floor(Math.random()*10);
10         function func()
11     {
12         var x = document.getElementById('txt');
13         x.value = "xxxxx";
14
15     }
16
17     function changeImg(id)
18     {
19         document.getElementById(id).src = "XAAB8584.JPG";
20
21     }
22
23     </script>
24 </head>
25 <body onload="func()">
26     <!--<input type="button" id="b1" onclick="func()" value="start" /-->
27     <input type="text" id="txt" value="abc" />
28     <input type="button" id="b1" onclick="changeImg('img1')"
29         value="changeImg" />
30     <br />
31     <input type="button" id="b2" onclick="changeImg('img2')"
32         value="changeImg" />
33     <br />
34     
35     
36     <!--<script src="JavaScript.js"></script>-->
37 </body>
38 </html>
```

מערך הוא מילוי מסודר המשמר בתוכו **אוסף של איברים**. כל חבר באוסף נקרא **איבר**. איברים במערך יכולים להיות מכל טיפוס נתונים כולל אובייקטים או אף מערכים אחרים. לכל איבר במערך יש **מספר ייחודי** רק לו נקרא **אינדקס**. בשונה משפטות תכונות רכבות אחרות, מערכים ב - **Javascript** אינם בעלי טיפוס נתונים מוגדר. משמעו הדבר היא שבתוך מערך אחד יכולים להיות איברים בעלי טיפוס נתונים שונה.

האיבר הראשון במערכים ב - **Javascript** יהיה בעל אינדקס שמספרו 0. המספר המקסימלי האפשרי של איברים במערך הוא: 4,294,967,295. חשוב לציין שמערכים ב - **Javascript** הם דינמיים ולכן כאשר יוצרים אותו אין צורך להגדיר מראש כמה איברים בתוכם.

יצירת מערכים

הדרך הפשוטה ביותר להגדרת מערך היא באמצעות הצבת משתנים מפורשת במערך כפי שניתן ללמוד מהדוגמאות הבאות.

דוגמאות קוד יצירה מערכים באמצעות הצבת איברים מפורשת בתוכם

```
var eArray = [] // מערך ריק
```

```
var i = [1, 2, 8, 7] // ארבעה איברים מספוריים
```

```
var ManyTypes = [11.132, false, "a"] // שלושה איברים מטיפוס נתונים שונים
```

הערה: שימו לב שאת האיברים למערך הגדרנו בתוך סוגרים מחובעים.

דרך אחרת ליצור מערך היא על ידי הגדרת משתנה מסוג מערך. נעשה זאת באופן הבא:

דוגמאות קוד יצירה מערכים באמצעות שימוש בקונסטרוקטור array

```
var eArray = new Array() // מערך ריק
```

```
var i = new Array(1, 2, 8, 7) // עם ארבעה איברים מספוריים
```

```
var ManyTypes = new array(11.132, false, "a") // מערך ובו שלושה איברים
```

```
מטיפוס נתונים שונים
```

הערה: שימו לב שאת האיברים למערכים הגדרנו בתוך סוגרים עגולים.

ישוב לציין שם גיב בטור הגדרה מפורשת, העשוה שימוש בקונסטרוקטור **array**, איבר מספרי אחד אך Javascript לא תרגום זאת כהצבת איבר בטור מערך אלא כהגדירה מראש של אורך המערך, קרי של מספר האיברים שיופיעו מראש בזיכרון המחשב עבור המערך הנוכחי. להלן דוגמת קוד לכך:

דוגמה להגדרת אורך מערך ושימוש בקונסטרוקטור array

```
var arr = new array(3) // מערך ובו הקצת מקום לשולשה איברים
```

בהגדרה מעין זו נשתמש במקרה שהוא יוזע מראש כמה איברים צפויים לאכלס אותו המערך כולו.

גישה לאיברים בתוך מערך

כל איבר במערך ישנו אינדקס המציין את מקומו בטור המערך. שפת Javascript מאפשרת לנו לגשת לאיבר בטור מערך פשוט על ידי שימוש באופרטור **[]** וציין האינדקס של האיבר אליו ברצוננו לגשת. באמצעות הסוגריים המרובעים המשמשים כאופרטור ב - **Javascript** יוכל לגשת לכל איבר במערך ולקרוא מה הערך המצו依 בו וכן לשנות את הערך המצו依 בו.

בדוגמה הבא נוצר מערך ובו שלושה איברים: 1, true - "Hello". לאחר מכן נראה כיצד נוכל לקרוא מה הערכים שבכל אחד משלושת האיברים שבמערך:

```
var a = [1, true, "Hello"] // צירת המערך והצבת הערכים בתוך המערך
var val1 = a(0); // val1 = 1
var val2 = a(1); // val2 = true
var val3 = a(2); // val3 = "Hello"
```

דוגמאות קוד לכתיבת ערכים בתוך מערך

```
var a = new array(3) // יצרת מערך ובו שלושה איברים ריקים
a(0) = 1; // נכתב את הערך 1 לאיבר הראשון במערך
var b = true; // נגידיר משתנה בוליאני ונציב בתוכו ערךאמת
a(1) = b; // נציב את ערכו של משתנה בתוך האיבר השני במערך
var i = 1; // נגידיר משתנה ובו הערך 1
a(i + 1) = "Hello"; // נכתוב ערך באיבר השלישי במערך
```

הוספת איברים למערך

הדרך הפשטית ביותר להוסף איברים למערך היא על ידי הצבת ערכים חדשים לתוך איברים חדשים הנוצרים על ידי שימוש במספר האינדקס שלהם. בדוגמה הבאה למשל ניצור מערך ריק ולאחר מכן נציב לתוכו ערכים. כל הצבה לאינדקס חדש תיצור איבר חדש במערך:

```
arr = [] // נגידיר מערך חדש ריק
arr[0] = 1; // a[0] נוסיף למערך איבר חדש
arr[1] = true; // a[1] נוסיף עוד איבר
arr[2] = "Hello"; // a[2] נוסיף עוד איבר
```

הוספת איבר חדש לסוף המערך

כדי להוסף איבר חדש שימוקם בסוף המערך ביצילתו להשתמש בשתי מתודות. האחת היא המתודה `push` שמטטרת היא להוסף איבר חדש בסוף מערך. להלן דוגמת קוד לשימוש במתודה `push`:

```
a = [] // נגידיר מערך חדש ריק
a.push(1) // a[0] נוסיף איבר חדש לסוף המערך
a = [1] // כך ייראה המערך שלנו לאחר הוספה האיבר
a.push(true, "Hello") // a[1] נוסיף עוד שני איברים
a = [1, true, "Hello"] // כ"ל ייראה המערך שלנו לאחר הוספה שני האיברים הנוספים
הדרך להוספה איבר חדש בסוף מערך היא פשוט למצואו מארך המערך ולוחץ ערך באינדקס השווה לארך המערך. כך יוצר לנו איבר חדש שימוקם בסוף המערך. כדי למצוא את אורך המערך נשתמש במתודה length:
```

```
a = [1, true]; // נגידיר מערך חדש ובו שני איברים
a(a.length) = "Hello" // ניצור איבר חדש בסוף המערך
a[1, true, "Hello"] // כך ייראה המערך שלנו לאחר הוספה האיבר לסוף
הביטוי a.length בדוגמה למעלה יחזיר את הערך 2. ערך זה מייצג את מספר האיברים במערך. היה ווב Javascript האינדקס של האיבר הראשון במערך הוא 0 אז במערך שבודגמה למעלה האינדקס של האיבר האחרון הוא 1. אי כן, הצבת איבר באינדקס השווה לארך המערך כמו כיצירת אינדקס חדש אחד לאחר האינדקס האחרון הקיימ.
```

הוספת איברים לכל מקום בתוך מערך באמצעות המתודה splice

המתודה splice מאפשרת להווסף איברים חדשים לתוך מערך בכל מקום שבו נרצה בכך. מתודה זו מאפשרת גם למחוק איזה איברים שנרצה (על אף שבבנין הבא במדריך זה). המתודה splice מקבלת שני ארגומנטים: הארגומנט הראשון מצין את המקום המדויק במערך שבו תבוצע פעולה הוספה. מכך. הארגומנט השני חשוב דווקא לצורך מחיקת איברים והוא מצין את מספר האיברים מנוקדת ההתחלת שאותם ברצונו למחוק. לאחר שני ארגומנטים אלה ניתן לציין רשימה של איברים שברצוננו להכניס במקומות המצוין במערך. לאחר ביצוע פעולה בערטת splice כל האיברים במערך מקבלים אינדקסים התואימים את מקומם במערך. כך שבמקרה שהתווסף איברים יתעדכו כל האינדקסים של האיברים שלאחר האיברים החדש.

splice מחזירה ערך. הערך המוחזר הוא מערך ובו רשימת האיברים שנמחקה מן המערך שלנו. במידה ולא בוצעה פעולה מחיקה מוחזר על ידי המתודה מערך ריק.

להלן דוגמאות קוד להוספת איברים לתוך מערכים באמצעות המתודה splice:

```
נדיר מערך ובו חמישה איברים // ;
var a = [1,2,3,4,5];
a.splice(2,0,'x','y'); // Returns []; a is [1,2,'x','y',3,4,5]
a.splice(2,2,[1,2],3); // Returns ['x','y']; a is [1,2,[1,2],3,3,4,5]
```

מחיקת איברים ממערך

נית להשתמש במתודה delete כדי למחוק איבר במערך. מתודה זו תגרום למחיקת האיבר אך לא לשינוי אורך המערך. לעומת זאת אם נתנו לנו מערך בין עשרה איברים שהאיבר האחרון שבו הוא בעל אינדקס 9 אזי מחיקת האיבר הרביעי למשל לא תגרום לשינוי האינדקסים של האיברים שלאחר האיבר שנמחק. להלן דוגמת קוד לשימוש במתודה delete.

```
נדיר מערך ובו שלושה איברים // ;
var a = [1, 2, 3];
delete a(1) //
```

מחיקת איברים מסוף מערך

יכול למחוק איברים מסוף המערך על ידי שינוי אורך המערך באמצעות המתודה `length`. בדוגמא הקוד הבאה נוצר מערך בן שלושה איברים ולאחר מכן נמחוק ממנו את שני האיברים האחרונים.

```
נדיר מערך בין 3 איברים // ;
var a = [1, 2, 3];
a.length=2; // אורך המערך הוא 2 במקומות 3
a = [1, 2] //
```

מחיקת איברים מכל מקום בתוך מערך באמצעות המתודה splice

המתודה splice מאפשרת למחוק איברים מכל מקום בתוך מערך: מתודה זו מאפשרת גם להווסף איזה איברים שנרצה (על אף שבבנין הבא במדריך זה). המתודה splice מקבלת שני ארגומנטים: הארגומנט הראשון מצין את המקום המדויק במערך שבו תבוצע פעולה המהינה / הוספה. הארגומנט השני מצין את מספר האיברים מנוקדת ההתחלת שאותם ברצוננו למחוק. לאחר ביצוע פעולה splice כל האיברים במערך מקבלים אינדקסים התואימים את מקומם במערך. כך שבמקרה שהתווסף איברים יתעדכו כל האינדקסים של האיברים שהו ממוקמים לאחר האיברים שנמחקו.

splice מחזירה ערך. הערך המוחזר הוא מערך ובו רשימת האיברים שנמחקה נון המערך שלנו. במידה ולא בוצעה פעולה מחיקה מוחזר על ידי המתודה מערך ריק.

להלן דוגמאות קוד למחיקת איברים בטור מערכים באמצעות המתודה splice:

```
var a = [1,2,3,4,5,6,7]; иудית מערך ובו שבעה איברים //  
a.splice(3); // Returns [4,5,6,7]; a is [1,2,3]  
a.splice(1,2); // Returns [2,3]; a is [1]
```

מיון ערכים אלפאנומריים בטור בעזרת המתודה sort

המתודה splice מאפשרת למיון מערכים מיון אלפאנומרי. כלומר, מותודה זו ביצורה הגלומית תתייחס לכל ערך בטור כל מחרוזת, גם אם מדובר למשל במספר.

דוגמה לשימוש במתודה sort לצורך מיון אלפאנומרי

```
var a=["World", "Israel", "Devschool", "Hello"]  
a.sort() //The outcome is: ["Devschool", "Hello", "Israel", "World"]
```

בדוגמה לעליה ראיינו כיצד המערך מוגןמי וסדר המין הוא מן האותיות הראשונות אל האחוריות. אם נרצה להפוך סדר זה כך שהמין יהיה מן האותיות האחרונות אל הראשונות, יוכל לעשות זאת באמצעות שימוש במתודה reverse.

דוגמה לשימוש במתודה reverse כדי להפוך את סדר המין של מערך

```
var a=["World", "Israel", "Devschool", "Hello"]  
a.sort() //The outcome is: ["Devschool", "Hello", "Israel", "World"]  
a.reverse //The outcome now is: ["World", "Israel", "Hello", "Devschool"]
```

מיון ערכים מספריים בעזרת המתודה sort

המתודה splice מאפשרת להעביר לה פונקציה מיון כארוגומנט. פונקציה זו מקבלת שני ארגומנטים ומוחזקה ערך מספרי כתשובה. הארגומנטים שמועברים לפונקציה הם ערכים שמשו את הפונקציה לצורך החלטה איזה מ-המספרים ימוין במקומות הראשוניים ואיזה במקומות הבאים. אם הערך המוחזר הוא שלילי אז הארגומנט הראשון שהועבר לפונקציה ימוין לפני הארגומנט השני, אם הערך המוחזר שווה ל-0 אז אין חשיבות לסדר המין של שני הארגומנטים, ככלומר הטעני שווים זה זהה מבחינת המין המבוקש.

בדוגמה הבאה ננאל את התוכנה המאפשרת לנו להעביר פונקציית מיון כארוגומנט למתודה splice כדי למיון מערך בעל ערכים מספריים.

מיון ערכים מספריים בטור בעזרת המתודה splice

```
var a=[23, 9, 6, 41];  
a.sort(function(a,b){return a - b});  
// The array now is [6, 9, 23, 41]
```

בדוגמה לעליה החסרים את הארגומנט הראשון מן השני. אם הארגומנט הראשון גדול יותר מן השני אז יוחזר ערך חיובי וללא הארגומנט השני יומוקם לפני הארגומנט הראשון. אם הארגומנט הראשון קטן מן הארגומנט השני אז הערך שיוחזר מחיסור שני הארגומנטים הוא שלילי וללא המספר הראשון יומוקם לפני המספר השני. אם הערך שיוחזר הוא אף משמעות הדבר שני הארגומנטים שוויים זה זהה וכן אין חשיבות לסדר המין.

כמובן שהפונקציה שכתבו בדוגמה الأخيرة תמיין את המספרים המועברים אליה בסדר עולה, ככלומר מן הקטן אל הגדל. אם נרצה להפוך את סדר המין נוכל להשתמש במתודה reverse כפי שהדגימו בסעיף הקודם אולם יוכל לעשות זאת גם בעזרת שינוי הפונקציה המミニות באופן הבא.

מיון ערכים מספריים בטור בסדר יורד

```
var a=[23, 9, 6, 41];  
a.sort(function(a,b){return b - a});  
// The array now is [41, 23, 9, 6]
```

בדוגמה לעליה הפכנו את סדר החישוף בין שני המספרים את המספר הראשון מהשני. שינוי זה יגרום לכך שסדר המין של הפונקציה יתהפך יחסית לסדר המין של הפונקציה בדוגמה הקודמת.

מציאת אורך מחרוזת

כדי למצוא מה אורך של מחרוזת נתונה ביכולתנו להשתמש בתוכנה **length** של כל משתנה בתוכנה מסוג מחרוזת. תוכנה זו אינה מקבלת ערכים והיא מחזירה את אורך המחרוזת.

הצורה הכללית של התוכנה **length**

```
string.length
```

דוגמת קוד למציאת אורך מחרוזת

בדוגמה הבאה נציג מחרוזת בתוך משתנה. לאחר מכן נשתמש בתוכנה **length** כדי למצוא מה אורך. לבסוף נכנו במספרם שלם את אורךה של המחרוזת:

```
<html>
<body>
<script type="text/javascript">
var str = "שלום טוזם";
document.write(str.length);
</script>
</body>
</html>
```

במדריך זה עוסוק בדרכים למציאת מחרוזות אחת בתוך מחרוזת אחרת. בפרט עוסוק בMETHOD **indexof** המאפשר לנו זאת.

חיפוש מחרוזת אחת בתוך מחרוזת אחרת

על מנת למצוא את מיקומה הראשון של מחרוזת אחרת בתוך מחרוזת נשתמש בMETHOD **indexof()**. כפי שנitin להתרשם מדוגמא הקוד ש滥מה מתודה זו מקבלת שני ערכים:

1. המחרוזת לחיפוש

2. נקודת התחלה החיפוש בתוך המחרוזון שבה נבצע את החיפוש (אופציוני).

הערך שיוחזר יהיה המיקום הראשון שבו נמצא המחרוזת. אם לא נמצא מופיע של המחרוזת המבוקש יוחזר הערך -1.

הצורה הכללית של השימוש ב - **indexof**

```
string.indexOf(substring) // search from the first character
string.indexOf(substring, start) // The use of start is optional
```

דוגמת קוד לחיפוש מחרוזת בתוך מחרוזת

בדוגמה הבאה נציג את המחרוזת "Devschool Israel" בvariable **str**. לאחר מכן נשתמש בMETHOD **indexof** כדי לאתר בתוך מחרוזת זו את המחרוזת "Israel". את מיקומה נציג בתוך המשתנה **pos**. לאחר מכן נדפיס את מיקום המחרוזת במסמן שלהם.

```
<body>
<script type="text/javascript">
var str = "Devschool Israel";
var pos = str.indexOf("Israel");
document.write("The position of the word Israel is " + pos);
</script>
</body>
```

כדי להחליף חלק מחוזות במחוזות אחרים ניתן להשתמש בMETHOD replace(). מетод זה מאפשר לנו להחליף בקלות כל חלק מחוזות במחוזות אחרים. המתודה מקבלת שני ערכים: הערך הראשון יכול את חלק מחוזות שברצוננו להחליף ואילו הערך השני יכול את המחזוזה שברצוננו שתתפס את מקומו של חלק מחוזות המוחלף.

דוגמת קוד להחלפת טקסט בתוך מחוזות

דוגמה הבאה נציג את המחזוזה "Hello World" במשתנה 1.str. לאחר מכן נשתמש בMETHOD replace כדי להחליף את המילה "World" במילה "Israel". את התוצאה נציג בתוך המשתנה2.str. לאחר מכן נדפיס את המחזוזה החדשה שנוצרה לאחר החלפה.

```
<html>
<body>
<script type="text/javascript">
var str1="Hello world!";
var str2=str1.replace("World", "Israel");
document.write(str);
</script></body>
</html>
```

פילוח וחיתוך מחוזות בשפת Javascript

במדריך זה עוסוק בזרכים לחיתוך ופילוח מחוזות. בפרט עוסוק בMETHOD slice המאפשר לבצע פעולות אלה בקלות וביעילות.

פילוח וחיתוך מחוזות

באמצעות המתודה slice ניתן לחותוך חלק מתוך מחוזות. המתודה מקבלת שני ערכים: הערך הראשון יכול את המיקום שמנגנו נרצה להתחיל בפיולוח ואילו הערך השני יכול את המיקום שבו יופסק הפילוח. המתודה מחזירה את תת המחזוזה שאותה חתכנו מחוזות המקיים.

הצורה הכללית של המתודה slice

```
string.slice(start, end)
```

דוגמת קוד לפילוח וחיתוך מחוזות

```
var s = "abcdefg";
s.slice(0,3) // Returns "abc"
s.slice(2,5) // Returns "cde"
s.slice(4) // Returns "efg"
s.slice(3,-1) // Returns "def"
s.slice(3,-2) // Returns "de"
```

חשוב: כפי שבירכלתכם להתרשם slice יכול לקבל ערכים שליליים כארוגומנט השני.

תאריכים

ב - Javascript, כאשר נרצה לעבוד עם תאריכים או ליצור משתנים שמחזקים בתוכם תאריכים נצטרך לעבוד עם האובייקט Date המוגנה בשפה. את האובייקט תאריך (Date Object) נוצר באמצעות (new Date()) שמודגם למטה.

מרגע שamo יוצרים את אובייקט התאריך (Date Object) ביכולתנו להפעיל עליו מספר רב של מетодות (methods) לצורך הצבת ערכים בתוכו, שילפת ערכים מןנו וכן כדי לבצע חישובים והמרהות מטיפוס נתונים אחד לשני.

יצירת אובייקט תאריך (Date Object)

להלן מספר דוגמאות לשימוש בקונסטרוקטור () Date כדי ליצור אובייקט מסווג תאריך:

1. new Date() התאריך הנוכחי של פי שעון המחשב //
2. new Date(milliseconds) מספר תשיריות השנה מתחילה שנה 1970 //
3. new Date(datestring) מהרוצת המציגת תאריך בפורמט המקובל בשפה //
4. new Date(year,month,date[,hour,minute,second,millisecond]) [] יצירת תאריך ושם //

הערה: הפורמטים שבתוך הסוגרים הם תמיד אופציונליים ולא חובה להשתמש בהם.

1. בדוגמה הראשונה ניצור אובייקט מסווג תאריך ובו ערך התאריך הנוכחי על פי שעון המחשב;

2. בדוגמה השנייה ניצור אובייקט שהתאריך שייחס יהיה זה שיועבר לו באלפיות השנה החל מה - 1/1/1970 .

3. בדוגמה השלישי נבהיר ארוגמנט מסווג מחוזת בפורמט המקובל על המונודה (Date.parse()).

4. בדוגמה הרביעית ניצור תאריך על פי הארגומנטים שיועברו. הארגומנטים מייצגים את ערךם של כל רכיבי התאריך והזמן המרכיבים תאריך חוקי בשפת Javascript.

כאשר משווים תאריכים ב - Javascript חשוב לציין שהאופרטור == מחזיר ערך אמת רק אם שני צדי המשוואה אינם משווים (תאריכים ≠ אותו אובייקט ותאריך). שימושות הדבר היא שאם יש לנו שני אובייקטי תאריך נפרדים תוצאה השוואתם באמצעות == תחזיר ערך שקר גם אם התאריך והשעה בהם זהים לחלווטן. לדוגמה הקוד הבא לא יחזיר לנו ערך אמת למרות שהתאריךים זהים:

```
var a = new Date("1/1/2013");
var b = new Date("1/1/2013");
document.write(a == b);
// Output: false
```

חשוב גם לדעת שככל שהשוואה בין תאריכים מתיחסת לא רק ליום חדש ושנה אלא גם לזמן. כך לא אוחת קורה שלמרות שהתאריך הקלינדרי זהה ההשוואה תחזיר ערך שקר מושם שהזמן בשני התאריכים אינו זהה. כך למשל, ערכו של התאריך שמחזירה המוגדרה () Date () יהיה שונה מעריך התאריך שיוחזר על ידי שימוש ב - ()now. Date משwon שעגת התאריך ב - ()now ()now:00 והוא השעה ב - ()now ()now:00 היא השעה הנוכחית על פי שעון המחשב.

בדוגמה הבאה הלקוצה מ- [MSDN](#) נשווה שני תאריכים:

```
// Get the current date at midnight.
var now = new Date();
var todayAtMidn = new Date(now.getFullYear(), now.getMonth(), now.getDate());

// Set specificDate to a specified date at midnight.
var specificDate = new Date("9/21/2009");

// Compare the two dates by comparing the millisecond representations.
if (todayAtMidn.getTime() == specificDate.getTime())
{
    document.write("Same");
}
else
{
    document.write("Different");
}
```

בדוגמה לעליה ביצמנו את ההשוואה של שני התאריכים על ידי שימוש בメונודה `getTime()`. מוגדר זו מוחזירה את מספר אלףות השנה החל מהתאריך 1.1.1970 בבחוץ ועד לתאריך המועבר כารוגומנט למתודה.

הפחתת תאריך מהתאריך

```
var bday = new Date("2/22/1964");
var thisday = new Date();
var hefresh = thisday.getFullYear() - bday.getFullYear();

// Change the year in bday to the current year.
bday.setFullYear(thisday.getFullYear());
```

```
// If the user's birthday has not occurred yet this year, subtract 1.
```

```
if (thisday < bday)
```

```
{
```

```
    hefresh--;
}
```

```
document.write(hefresh);
```

בדוגמה הקוד שילבала יצרנו שני אובייקטים מסוג תאריך. אחד מחזק את תאריך ים הולדת ואילו השני את התאריך הנוכחי. לאחר מכן הפחתנו מהשנה שבתאריך הנוכחי את השנה שבתאריך ים ההולדת. התוצאה שהתקבלה היא גילו של האדם, אלא אם כן טרם הגיע ים הולדתו בשנה זו. במקרה זה יש להפחית שנה אחת מהנתוצאה שהתקבלה, אחרת יש להשאיר את הנתוצאה כפי שהיא.

האובייקט `Date` ב- `Javascript` מאפשר לנו להפחית ימים, חודשים או שנים ממועדן או להוסיף ערכיהם אלה לתאריך. זאת באמצעות המתודת השונות המאפשרת לעדכן את הערכים השונים מהם מורכב תאריך.

הוספה / הפקת ימים לתאריך

בדוגמה הבאה נוסיף יום אחד לתאריך קיים:

```
var myDate = new Date("1/1/2013");
myDate.setDate(myDate.getDate() + 1);
document.write(myDate);
// Output: Wed JAN 02 00:00:00 PST 2013
```

הוספה / הפקת חודשים מתאריך

בדוגמה הבאה נפחת חודש אחד מתאריך קיים:

```
צור תאריך חדש // הפחת חודש מהתאריך שיצרנו //
myDate.setMonth(myDate.getMonth() - 1);
document.write(myDate); // הדפס את התאריך החדש לאחר ההפחתה
// Output: Sat Dec 01 2012 00:00:00
```

הוספה / הפקת שנים לתאריך

בדוגמה הבאה נוסיף שנה לתאריך:

```
var myDate = new Date("1/1/2013");
myDate.setYear(myDate.getFullYear() + 1);
document.write(myDate);
// Output: Wed Jan 01 2014 00:00:00
```

הMETHODS (Methods) שהן חלק מן האובייקט תאריך (Date Object) נחלקות לשלווש מחלקות עיקריות:

1. **חילוץ ערכים** - METHODS שתפקידן לחילוץ ערך מתאריך ותאריך נתון, כגון היום בתאריך או החודש או השנה וכו'.
2. **עדכון ערכים** - METHODS שתפקידן להכניס (לשנות) ערך בתאריך כגון היום, השעה, החודש וכו'.
3. **המרה** - METHODS שתפקידן להמיר את נתונים התאריך מטיפוס נתונים אחד לשנהו.

METHODS תאריך (Date Methods) המחלצות ערכים מטוק תאריכים

METHOD	תיאור
getDate	החזרת היום בחודש (1 - 31) בתאריך נתון.
getDay	החזרת היום בשבוע (0 - 6) של תאריך נתון.
getFullYear	החזרת שנת התאריך (בפורמט מלא של 4 ספרות).
getHours	החזרת השעה (0 - 23) בתאריך נתון.
getMilliseconds	מחזיר את אלפיות השניה (0 - 999) בתאריך המזמין.
getMinutes	מחזיר את מספר הדקה (0 - 59) בתאריך נתון.
getMonth	מחזיר את מספר החודש בתאריך (0 - 11) בתאריך נתון.
getSeconds	מחזיר את מספר השניה בתאריך (0 - 59) בתאריך נתון.

מחזיר את מספר אלףות השנייה שUbero מהתאריך 1.1.1970 ועד לתאריך הנוכחי.

getTime

מחזיר את הפרש הזמן בין הזמן המקומי ל - GMT. יש לשים לב שהערך חזר בזמן הפוך מהמצופה. ככלומר למשך שאנו ב - GMT+3 למשך השער שיווזר יהיה -3. זאת משום שהיא שמתוודה עשוה הוא לחסר את זמן ה - GMT מהזמן המקומי ולא לורוף.

getTimezoneOffset

מחזיר את היום (1 - 31) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCDate

מחזיר את היום (0 - 6) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCDay

מחזיר את השנה המלאה (4 ספרות) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCFullYear

מחזיר את השעה (0 - 23) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCHours

מחזיר את מספר אלףות השנה (0 - 999) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCMilliseconds

מחזיר את מספר הדקות (0 - 59) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCMinutes

מחזיר את מספר החודש (0 - 11) בתאריך מציין על פי הזמן האוניברסלי.

getUTCMonth

מחזיר את מספר השניות (0 - 59) בתאריך המצוין על פי הזמן האוניברסלי.

getUTCSeconds

מחזיר את מספר אלףות השנה (2 עד 3 ספרות). אין להשתמש יותר במתודה זו משום שהיא מהשפה. במקרה זאת יש להשתמש במתודה `getFullYear`.

getYear

מחזיר את הערך הפרימיטיבי של תאריך נתון.

valueOf

מתודות ותאריך (Date Methods) להמרת טיפוסי נתונים

תיאור	מתודה
עדכן היום בחודש (1 - 31) בתאריך נתון.	<code> setDate</code>
עדכן שנת התאריך (בפורמט מלא של 4 ספרות).	<code> setFullYear</code>
עדכן השעה (0 - 23) שנתאריך נתון.	<code> setHours</code>
עדכן מספר אלףות השנה (0 - 999) בתאריך המצוין.	<code> setMilliseconds</code>
עדכן את מספר הדקה (0 - 59) בתאריך נתון.	<code> setMinutes</code>
עדכן את מספר החודש בתאריך (0 - 11) בתאריך נתון.	<code> setMonth</code>
עדכן את מספר השנה בתאריך (0 - 59) כחאריך נתון.	<code> setSeconds</code>
קבע את התאריך בעורcht עדכונומספר אלףות השנייה שUbero מהתאריך 1.1.1970 ועד לתאריך הרצוי.	<code> setTime</code>
עדכן את היום (1 - 31) בתאריך המצוין על פי הזמן האוניברסלי.	<code> setUTCDate</code>
עדכן את השנה המלאה (4 ספרות) בתאריך המצוין על פי הזמן האוניברסלי.	<code> setUTCFullYear</code>
עדכן את השעה (0 - 23) בתאריך המצוין על פי הזמן האוניבурсלי.	<code> setUTCHours</code>
עדכן את מספר אלףות השנה (0 - 999) בתאריך המצוין על פי הזמן האוניברסלי.	<code> setUTCMilliseconds</code>
עדכן את מספר הדקות (0 - 59) בתאריך המצוין על פי הזמן האוניברסלי.	<code> setUTCMinutes</code>
עדכן את מספר החודש (0 - 11) בתאריך מציין על פי הזמן האוניברסלי.	<code> setUTCMonth</code>
עדכן את מטף השנה (0 - 59) בתאריך המצוין על פי הזמן האוניברסלי.	<code> setUTCSeconds</code>
עדכן את מספר השנה (2 עד 3 ספרות). אין להשתמש יותר במתודה זו משום שהיא מהשפה. במקרה זאת יש להשתמש במתודה <code>getFullYear</code> .	<code> setYear</code>

מתודות תאריך (Date Methods) למרת תאריכים ושעות לטיפוסי נתונים (Data Types) אחרים

תיאור	מתודה
המרת תאריך היום לפורת קרי (באנגלית) כגון: Fri Feb 22 1964	<code> toString</code>
המרת אובייקט תאריך למהירות בפורט ISO ISO例如: T14:48:00.000Z2012-10-05	<code> toISOString</code>
ממיר תאריך כערך שיווז בפורט JSON. נමך החל מגרסת Javascript 1.8.5. ממיר תאריך כערך שיווז בפורט JSON. נתמך החל מגרסה 1.8.5	<code> toJSON</code>

בוטל מהשפה ואין להשתמש יותר. מקום מתודה זו ניתן להשתמש ב - .toUTCString	toGMTString
מחזיר את התאריך כמחרוזת בפורמט המקביל למקום.	toLocaleDateString
מחזיר את חלק הזמן בתאריך כמחרוזת בפורמט מקומי.	toLocaleTimeString
ממיר אובייקט תאריך למחרוזת.	toString
ממיר את חלק הזמן בתוך תאריך למחרוזת.	toTimeString
ממיר תאריך למחרוזת על פי הזמן האוניברסלי	toUTCString

```

1 <html dir="rtl">
2
3 <head>
4 <title>Math</title>
5 </head>
6
7 <body>
8   <script language="javascript">
9
10  var x=Math.abs(-5); //x=5
11
12  x=Math.ceil(4.12); //x=5
13
14  x=Math.floor(4.12); //x=4
15
16  x=Math.round(4.12); //x=4
17
18  x=Math.round(4.55); //x=5
19
20  x=Math.max(4,5); //x=5
21
22  x=Math.min(4,5); //x=4
23
24  x=Math.pow(2,3); //x=8
25
26  x=Math.sqrt(9); //x=3
27
28  x=Math.random(); //0-1
29
30  x=Math.floor(Math.random()*10+1); //0-10
31  //Math.floor(copies the integer part of the number)
32
33
34  var radius = 12;
35  var area = Math.PI * radius * radius; //3.14159...*144
36  document.write(" שטחו של מעגל שרדיוס 12 הוא " + area + "<br>"); // 452.3893421169302
37
38  document.write(Math.round(area*1000)/1000); //452.389
39  //copies the first three digits after the decimal point
40
41 </script>
42
43 </body>
44 </html>
45

```

מחזירה את הערך המוחלט של מספר -
 מעגלת מספר כלפי מעלה אל המספר השלם הקרוב ביותר -
 מעגלת מספר כלפי מטה אל המספר השלם הקרוב ביותר -
 מעגלת מספר למספר השלם הקרוב ביותר -
 מחזירה את הערך הגדול מביניהם
 מחזירה את הערך הקטן מביניהם
 מחזירה את ההעלאה בחזקה של המספר הראשון במספר השני -
 מחזירה את השורש של המספר -
 מוחללת מספר אקראי - המטרפ הרוחז הוא שבר עשרוני במלול -
 התוצאה מספר אקראי בין 1-10 //10
 הכפלת המספר האקראי ב-10, והפיקתו למספר שלם בין 1-10 באמצעות השיטה
 הנקודה ב-1000 כדי להוציא את הנקודה וعشרונית, עוגול אל המספר השלם הקרוב. לבסוף מוחלק העוגר ב-1000 כדי להוציא את הנקודה //

```

1 <script language="javascript">
2
3     var d1=new Date(); // מיצג את נקודת הזמן הנוכחי
4     document.write(d1+"<br>"); //Mon Dec 5 22:35:58 UTC+0200 2005
5
6     // בניית שמהלך ערכים מסתירים את השנה , החודש, היום, השעה, מספר הדקות את מס' השניות
7     var d2=new Date(2006,2,28,10,45,3);
8     document.write(d2+"<br>"); //Tue Mar 28 10:45:03 UTC+0200 2006
9
10    // בניית שמקבל מחרוזת תווים שמתחילה נקודת זמן מסוימת, התבנית של פיה מחרוזת הרווחים צריכה להיות://
11    var d3=new Date("March 28,2006,10:45");
12    document.write(d3+"<br>"); //Tue Mar 28 10:45:00 UTC+0200 2006
13
14    // כל נתון שימושת מבין נתונים השעה יקבל את הערך אפס/
15
16
17    var ttt=d1.getTime(); //1133814958642
18    var d4=new Date(ttt); //1/1/1970 בבחוץ הלילה
19    document.write(d4+"<br>"); //Mon Dec 5 22:35:58 UTC+0200 2005
20
21
22    //הזרת ערכים מ-Date
23    //get
24    // Mon Dec 5 2005 22:35:58 הדוגמאות לתאריך
25
26    document.write(d1.getDate()+"<br>"); //5 מחרירה את היום בחודש -
27
28    document.write(d1.getMonth()+1)+"<br>"); //12 ( טווח הערכים 0-11) מחרירה את החודש -
29
30    document.write(d1.getYear()+"<br>"); //2005 מחרירה את השנה (2 ספרות)
31
32    document.write(d1.getDay()+1)+"<br>"); //2 מחרירה את מספר היום בשבוע -
33
34    document.write(d1.getHours)+"<br>"); //22 מחרירה את השעה -
35
36    document.write(d1.getMinutes)+"<br>"); //35 מחרירה את מספר הדקות -
37
38    document.write(d1.getSeconds)+"<br>"); //58 מחרירה את מספר השניות -
39
40
41    var dd=new Date(1999,2,28);
42    document.write(dd.getFullYear()+"<br>"); //99
43    document.write(dd.getFullYear)+"<br>"); //1999
44
45    //-----
46
47    document.write(d1.toDateString()+"<br>"); //Mon Dec 5 2005
48
49    document.write(d1.toTimeString)+"<br>"); //22:35:58 UTC+0200
50    var s=d1.toString();
51    alert(typeof(s)+"---"+s);
52
53
54    //-----
55
56    //Date קביעת ערכים ב-
57    //set
58
59    // קובע את הזמן והתאריך על ידי מספר אלףיות השנה אשר חלפו מ-1 בינואר 1970 - setTime()
60    var d5=new Date(); //
61    document.write(d5+"<br>"); //Sat Dec 10 21:20:06 UTC+0200 2005 קביעת התאריך לעוד 5 ימים +
62    d5.setTime(d5.getTime()+(1000*60*60*24*5)); //+
63    document.write(d5+"<br>"); //Thu Dec 15 21:20:06 UTC+0200 2005
64
65    var date=new Date();
66    date.setDate(20); //קובעת את היום בחודש//
67    date.setMonth(2); //קובעת את החודש (חנוכה השנה)
68    date.setFullYear(2005); //קובעת את השנה//
69    date.setHours(8); //קובעת את השעה//
70    date.setMinutes(30); //קובעת את הדקות//
71    date.setSeconds(5); //קובעת את השניות//
72
73    document.write(date); //Sun Mar 20 08:30:05 UTC+0200 2005
74    //Date//
75 </script>

```

```

1 <html dir="rtl">
2 <head>
3   <title> Array מערך </title>
4
5   <script language="javascript">
6     function addName()
7     {
8       alert(nameArr.length);
9       // ניתן להוסיף ערך מעדכן את התוכנה
10      length;
11      nameArr[nameArr.length]=prompt("הכנס שם");
12      alert(nameArr.length);
13      nameArr[nameArr.length+5]="tttt"; // מגדיל את המערך בערך מקומות ושם ערך במקומות החמישי שנוסף
14      alert(nameArr[nameArr.length-2]); //undefined
15      //הוכנה גודל התעדכונה
16      alert(nameArr); //פיסיקום רק פסיקום
17      //alert(nameArr.shift());
18      nameArr.unshift("tttt");
19
20    }
21
22    function delName(i)
23    { // nameArr.splice(i,2,"aaa","bbb");
24    alert(nameArr.length+ nameArr);
25    // אם יש 6 איברים כגון
26    // יוסרו האיברים ממקומות 2 ו 3
27    // nameArr.splice(2,2); //2 איברים
28    // alert(nameArr.length+ nameArr); //4aa bb ee ff
29
30    // יוסרו האיברים ממקומות 2 ו 3 ובמקומות יוכנסו xxx yyyy
31    // alert(nameArr.length+ nameArr); //6aa bb xxx yyyy ee ff
32
33    // אם יש 6 איברים כгון
34    // nameArr.splice(2,2,"xxx","yyy","zzzz"); //3 איברים והשאר דוחקן
35    // alert(nameArr.length+ nameArr); //6aa bb xxx yyyy zzzz ee ff
36
37    // לא מוחק איברים אך הוכניס וק את ההפתרומים הנוספים
38    // nameArr.splice(2,"xxx","yyy","zzzz"); //()
39    // alert(nameArr.length+ nameArr); //6aa bb xxx yyyy zzzz cc dd ee ff
40
41    // מוחיקת לפתרומת מוחיקה
42    // alert(nameArr.length+ nameArr); //6aa bb xxx yyyy zzzz cc dd ee ff
43
44  }
45
46  function printNames()
47  { alert(nameArr.join()); // שיטה מצטטת את ערכי המערך למחוזת אחת אורךה
48  // הערכים מופדרים ברוח
49  // var str=nameArr.join("-"); // הצבת המחרזת למשתנה
50  // alert(str);
51  }
52
53  function concatArr()
54  { var arr1=new Array(1,2,3,4,5), arr2=new Array(6,7,8);
55  // שרשור מערכיים
56  // var arr3=arr1.concat(arr2); // arr3=1,2,3,4,5,6,7,8
57  // alert(arr3.join());
58  // var arr4=new Array(9,10);
59  // arr3=arr1.concat(arr2,arr4); // arr3=1,2,3,4,5,6,7,8,9,10
60  // alert(arr3.join());
61  }
62
63  function reverseArr()
64  { var arr=new Array(1,2,3,4);
65  // alert(arr.join());
66  // arr.reverse(); //arr=4,3,2,1
67  // alert(arr.join());
68  }
69
70  </script>
71 </head>
72 <body>
73  <script language="javascript">
74
75  var daysArr=new Array("ראשון", "שני", "שלישי", "רביעי", "חמישי", "שישי", "שבת");

```

```

76    תכונה המכילה את מספר איברי המערך // length
77    document.write(daysArr[i]+"<br>");
78
79
80    var num;
81    num=prompt("3,"; (הכנס מספר שמות לקליטה)
82        var nameArr=new Array(num);
83        for(i=0;i<num;i++)
84            nameArr[i]=prompt(" " שם מסטר"+ (i+1));
85        for(i=0;i<nameArr.length;i++)
86            document.write(nameArr[i]+"<br>");
87    var arrint=new Array(1,2,3,4,5,6,7);
88    מאיבר0 עד איבר שלפני 5
89    arrint2=arrint.slice(1,5); // 2,3,4,5
90    alert(arrint2); // 2,3,4,5
91    arrint2=arrint.slice(3); // סוף המערך
92    alert(arrint2); // 4,5,6,7
93
94    ראייתי בספר שמחזיר אורך המערך החדש אך לא הדפיס //
95    //????? alert(x); //undefined
96    alert(arrint); // 20,19,18,1,2,3,4,5,6,7
97    x=arrint.shift(); // 20 הורח התא הראשוני
98    alert(x);
99    alert(arrint);
100 </script>
101 <form>
102     <input type="button" value="הוסף שם" onclick="addName()">
103     <input type="button" value="מחק שם" onclick="delName(2)">
104     <input type="button" value="הדפסה השמות" onclick="printNames()">
105     <input type="button" value="מיון המערך" onclick="nameArr.sort()">
106     <input type="button" value="שרשור מערכים" onclick="concatArr()">
107     <input type="button" value="היפוך סדר המערך" onclick="reverseArr()">
108 </form>
109 </body>
110 </html>

```

```
1 <html>
2 <head>
3 </head>
4
5 <body>
6
7 <h1 align="center">String - אובייקט המחראות </h1>
8
9 <script language="javascript">
10    var str1=new String("Java"), str2=new String("Script"), str3, str4;
11    str3=str1+str2; //str3=JavaScript
12    alert(str3); // JavaScript הפלט
13    str4=str1.concat(str2); //str4=JavaScript
14    alert(str4.length); //10המחראות
15    var c1=str4.charAt(0); //c1=J
16    var c2=str4.charAt(9); //c1=t
17    alert(c1+ " " + c2);
18    str3=str4.substring(2,7); //str3=vaScr
19    alert(str3);
20    c1=str4.indexOf("Script"); //c1=4
21    alert(c1);
22    c1=str4.indexOf("a",2); //c1=3;
23    alert(c1);
24    c1=str4.lastIndexOf("a"); //c1=3;
25    alert(c1);
26    alert(str4.split("")[0]+dddd);
27    var arr1=str4.split(); //arr1=('J','a','v','a','S','c','r','i','p','t') arr1.length=10
28    alert(arr1.join()+"kj1");
29    alert(arr1.length);
30    var str5=new String("This is a new String");
31    var arr2=str5.split(" "); //arr2=("This","is","a","new","String") arr2.length=5
32    alert(str5.split(" ").length); //5 הפלט
33    alert(str5.split(" ")[0]); //This הפלט
34    str5=str4.toUpperCase(); //str5=JAVASCRIPT
35    alert(str5);
36    var string1=new String("abcdef");
37    document.write("<br/>" + string1.sub);
38    //alert(string1.strike);
39
40 </script>
41
42 </body>
43 </html>
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script type="text/javascript">
7          var i = 0;
8          function f()
9          {
10              document.getElementById('txt').value = ++i;
11          }
12          var x;
13          function fstart()
14          {
15              x=setInterval("f()", 500);
16          }
17          function fstop() {
18              clearInterval(x);
19          }
20      </script>
21  </head>
22  <body>
23  <input type="text" id="txt" />
24      <br />
25      <input type="button" onclick="fstart()" value="start" />
26      <input type="button" onclick="fstop()" value="stop" />
27  </body>
28  </html>
```

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title></title>
6      <script type="text/javascript">
7          function f() {
8
9              console.log(document.getElementById('div1').innerHTML);
10         }
11         function f2()
12         {
13             if (document.getElementById('b1') == null) {
14                 var x = prompt("press", "ok")
15                 document.getElementById('div1').innerHTML += '<input'
16                 type="button" id="b1" value="' + x + '" />';
17             }
18             var s = "abc";
19             document.getElementById('div1').innerHTML += s.link
20                 ("HtmlPage2.html");
21             document.getElementById('div1').innerHTML += s.strike();
22
23             for (var i = "a".charCodeAt(0); i <= 'z'.charCodeAt(0); i++)
24                 console.log(String.fromCharCode(i));
25         }
26     </script>
27 </head>
28 <body onload="f()">
29     <div id="div1">
30         <input type="text" id="txt1" />
31     </div>
32     <input type="button" id="b2" value="start" onclick="f2()" />
33 </body>
34 </html>
```

airyuim ב - Javascript

airyuim ומטפלי airyuim, הוסף מטפלי airyuim, מחייבת מטפלי airyuim ודברים חשובים לשדים לב אליהם כשרושים מטפל לairyu .

מהו airyu?

از לחדים מביננו, airyu הוא בעצם מהו שמתרכש. airyu לרוב מתרחש על ידי המשתמש, אך לא תמיד (כמו שנראה בהמשך) airyu לדוגמה: לחיצה על העכבר, תזוזה של העכבר, לחיצה על מקש במקלדת וכו'. כל אלה הם airyuim.

דוגמאות לairyuim

airyu	
click	כשמקליקים על אלמנט
mouseover / mouseout	כשנכנסים או יוצאים עם העכבר מאלמנט
keyup / keydown	כשלוחצים על כפתור במקלדת
focus / blur	כשנכנסים או יוצאים משדה בטופו
select	-CBSOBHHRIM URK M select list
submit	כשמגיישים טופו
load	כשהדף סיים להטען
unload	כשייצאים מדף

מה עושים עם airyuim האלה?

לפעמים נרצה לעשות משהו מסוים דזוקא כשמשהו קורה. לדוגמה, נרצה להקפייז הודעה כאשר המשתמש ילחץ על כפתור מסוים באתר שלנו. בדיקן נוכנים airyuim. כדי לישם את הדוגמה שלנו אנחנו נצטרך להשתמש באiryu click. click אונחנו נצטרך לתפוס את airyu ולדאוג שקוד מסוים (הקפצת הודעה במקירה שלנו) יפעל כאשר airyu מתרחש. אחת הביעות בהצמדת airyuim באופן הבא:

```
el.onclick = function() { alert('hello');}
```

היא שאם אני מצמיד שני airyuim זהים לאותו אלמנט (תחשבו על window.onload), אחד airyuim ידרס את השני.

לפיכך אנחנו צריכים להשתמש ב-`addEventListener.add`. מוגדר פונקציה שמאפשרת הוספה נוספת כמו וכמה אירועים לכל אלמנט.

אירועים ומטפלים אירועים

כדי להתחיל, נצטרך להבדיל בין האירועים לבין מטפל האירועים.

האירועים - הדברים שתרחשים כמו לחיצה על העכבר, לחיצה על כפתור במקלדת וכדומה.

מטפל האירועים - מה שבעצם מתרחש כשהאירוע מופעל, הדבר שמטפל באירוע.

חשוב להבין שלפעמים לאירוע אין מטפל אירוע, ובמקרה זה פשוט לא יקרה כלום שהאירוע יתרחש.

לפעמים גם המטפל הופיע קיימ. לדוגמה יש מטפל לאירוע `abc` כשהאירוע `abc` בכלל לא קיים. דבר זה קורה בדרך כלל כתוצאה של המתוכנת ואין סיבה לכתוב מטפלים לאירועים שלא קיימים.

מטפל אירוע נקרא גם **מארין לאירוע** (`event listener`) כי הוא בעצם "מצביע" ומחייב לפעול ברגע שהאירוע יתרחש.

טיפול באירועים

כדי לטפל באירוע אנחנו משתמש בפונקציה `addEventListener`.

הפונקציה `addEventListener` מקבלת שלושה פרמטרים (הרביעי אינו חלק מהסטנדרט)

- **type** - סוג אירוע. הכוונה היא ל `click`-`mouseover` של לחיצה `mouseover`, במקרה של מעבר עכבר וכדומה.

- **listener** - מטפל אירוע. כאן נעביר פונקציה שתתבצע כאשר אירוע יתרחש. הערך של `this` בפונקציה הוא האלמנט אליו הצמדנו את מטפל אירוע.

useCapture - אופציוני.

לדוגמא, אם נרצה להקפיז הודעה כאשר לוחצים על אלמנט מסוים, נציג לו מטפל לאירוע `click` בצורה הבאה:

```
var element = document.getElementById("ID"); // getElementById is just for example
element.addEventListener("click", function() {
    alert("Hello!");
});
```

הפרמטר `useCapture`

הפרמטר זה בא לטפל במקרים מסוימים מאוד.

נראה את הקוד הזה:

```
<div id="a">
  a
  <div id="b">
    b
  </div>
</div>
<script>
  document.getElementById("a").addEventListener("click", function () {
    alert("Need to be first!");
  });
  document.getElementById("b").addEventListener("click", function () {
    alert("Hi!");
  });
</script>
```

```
3 javascript-ב-ירועים
  });
</script>
```

המטרה שלנו כאן היא שיכש א-נלחץ (גם אם אלמנט בתוכו נלחץ "Need to be first", תמיד יוצג ראשון לפני כל מטפל אירוע אחר. אם נרים את הקוד ונלחץ על b נגלה שהה שזה לא מה שקרה. קודם מוקף Hi ורק אז Need to be first. וזה לא מה שאנו חצימים. למה זה קורה? כי ברגע שנחצינו לוחצים על b מופעלים קודם מטפל האירועים של b, ורק אז מופעלים מטפל האירועים של a כי האמת שgam לחצו על a (אמנם לא ישירות). כך ההיררכיה עובדת.

הפרמטר `useCapture` בא לפטור לבדוק את זה. אם נעביר לו `false`, נקבל את התוצאות שראינו מקודם. לעומת זאת, אם נעביר `true` אז מטפל האירוע שרשמנו זה עתה יפעל תמיד לפני כל מטפל אירוע של אחד מהאלמנטים שמתחתיו בעץ ה-DOM (כלומר אלמנטים שהם בתוכו).

חשוב מאד להבין: זה לא אומר שמטפל האירוע תמיד יפעל ראשון. אלא פשוט לפני כל מטפל האירועים של האלמנטים שבתוכו (מתחתיו בעץ היררכיה). וכך שאם יש אלמנט מעליו (אלמנט שהוא "אב" של האלמנט שלנו) אז מטפל האירועים של האב יפעל ראשון (כמובן אם גם הוא רשם עם `useCapture=true`)

בכה שאם נתקן את הקוד שלו:

```
<div id="a">
  a
  <div id="b">
    b
  </div>
</div>
<script>
  document.getElementById("a").addEventListener("click", function () {
    alert("Need to be first!");
  }, true);
  document.getElementById("b").addEventListener("click", function () {
    alert("Hi!");
  });
</script>
```

עכשיו `Need to be first` יהיה ראשון כナルחץ על b.

כמה מטפלים לאותו אירוע, אפשרי?

כמובן.

נראה את הקוד הבא:

```
<div id="a">
  a
</div>
<script>
  document.getElementById("a").addEventListener("click", function () {
    alert("first event listener");
  });
  document.getElementById("a").addEventListener("click", function () {
    alert("second event listener");
  });
</script>
```

רשכנו את שני מטפלים לאותו אירוע. מה יקרה כשהナルחץ על a?

נראה שנקבל event listener ראשון וevent listener שני מטפל האירועים הופעל. אפשר לראות שמי שנרשם לאירוע ראשון מופעל ראשון. כל הקודם זוכה. גם כאן useCapture משחק תפקיד, ואם ניתן למשהו true אז הוא יופעל לפני כל השאר (גם אם הוא נרשם אחרון לאירוע).

(element.onclick, inline javascript attachEvent) - מיעדים לדפנות ישנים.)

מחיקת מטפל אירועים

לפעמים לא נרצה עוד להקפיז הודעה למשתמש כשהוא לוחץ על הכפתור. מה נעשה אז? במקרה זה נצטרך למחוק את המטפל של האירוע click של הכפתור שמקפיז את הודעה. את זה נעשה באמצעות הפונקציה removeEventListener. הפונקציה מקבלת גם היא 3 פרמטרים: - type של האירוע שלו אנחנו מוחקים את המטפל. - listener של האירוע אותו אנחנו רוצים למחוק. (useCapture false חובה - (בעיר true אם המטפל אותו אנחנו מוחקים נרשם כ useCapture=false אם לא).

דוגמה שתמחיש את העניין:

```
<button id="a">a</button>
<button id="b">b</button>
<script>
function listener() {
  alert("clicked!");
}
document.getElementById("a").addEventListener("click", listener);
document.getElementById("b").addEventListener("click", function () {
  document.getElementById("a").removeEventListener("click", listener);
});
</script>
```

אפשר לראות כאן שכותבנו פונקציית listener פשוטה שמקפיiza הודעה ורשمنו את הפונקציה כמטפל לאירוע click של הכפתור a. רשمنו גם מטפל לאירוע click של הכפתור b שמוחק בעצם את המטפל listener של האירוע click של הכפתור a. נראה שגם אם אנחנו לוחצים על a אנחנו מקבלים הודעה. אבל אם נלחץ על b ואז נלחץ שוב על a, נפסיק לקבל הודעה. כי הלחיצה על b מחקה את המטפל של האירוע של a.

הפרמטר event

מטפל האירוע שלנו (event) מקבל גם פרמטר. הפרמטר הוא אובייקט מסווג Event. לאובייקט זהה יש הרבה מאפיינים ופונקציות והם משתנים בסוגים שונים של אירועים,

- האלמנט אליו רשםנו את האירוע (ומוביל לשימוש בזה ולא ב this-כי את this אפשר לשנות). currentTarget

- האלמנט המקורי ממנו הופעל האירוע. זה לא זהה לאלמנט אליו רשםנו את מטפל האירוע. במקרה שהצטט על אחד מהאלמנטים בתוך האלמנט אליו רשםנו את מטפל האירוע אז currentTarget יחזיר לנו את אלמנט האבא אליו רשםנו את מטפל האירוע, אבל target יחזיר את האלמנט שעליו לחצנו, שהוא לא בהכרח האלמנט אליו רשםנו את מטפל האירוע.

- מחזיר את סוג האירוע click, mouseover וכו'.

- מבטל את ההתנהגות הנורמלית של האירוע. preventDefault

```

5  אירוחים ב-Javascript
<a href="http://google.co.il" id="google">Google</a>
<script>
  document.getElementById("google").addEventListener("click", function (e) {
    e.preventDefault();
  });
</script>

```

כשנלחץ על ה קישור לא נ עבור ל גוגול כיון שהפעלנו את preventDefault שמנעה את התנהגות ברירת המחדל של האירוע על אותו אלמנט (במקרה שלנו - לחיצה על-link שתוביל לעמוד אחר).

- stopImmediatePropagation אם נפעיל את הפונקציה זו באחד מהמתפלים של אירוע מסוים, אז גם אם יש מתפלים נוספים לאותו אירוע, הם לא יפעלו באירוע הנוכחי.

דוגמה:

```

<button id="a">click</button>
<script>
  document.getElementById("a").addEventListener("click", function (e) {
    alert("Hello");
    e.stopImmediatePropagation();
  });
  document.getElementById("a").addEventListener("click", function () {
    alert("Hi!");
  });
</script>

```

אם נרים ונלחץ על הכפתור נגלה שלא קיבל את הודעה Hi אלא רק את Hello למה? כי אחרי שהקפינו את הודעה Hello הפעלנו את stopImmediatePropagation בעמם משאר מתפלי האירועים גם הם פועל.

שימוש לב: מתפל האירוע שמקפץ את הודעה Hi עדין רשום לאירוע. לא מחקנו אותו. ככה שאם לדוגמה נמחוק את מתפל האירוע הראשון אז אנחנו מקבלים הודעה Hi חשוב להבין את זה.

stopPropagation - מונע מהאירוע להמשיך "להתפשט" ולהפעיל מתפל אירועים באלמנטים אחרים במעלה עץ ה DOM-אלמנטים שהם "הורים" של האלמנט שהפעיל את אירוע.

```

<div id="a">
  a
  <div id="b">
    b
  </div>
</div>
<script>
  document.getElementById("a").addEventListener("click", function () {
    alert("Hello");
  });
  document.getElementById("b").addEventListener("click", function (e) {
    alert("Hi!");
    e.stopPropagation();
  });
</script>

```

אם נרים ונלחץ על ס נראת שאנחנו מקבלים את הודעה Hi אבל לא מקבלים את>Hello מה? כי ברגע של לחץ על b והקפינו את הודעה Hi הפעילו את stopPropagation שמנע אלמנטים במעלה עץ ה DOM- (אלמנט כמו a לדוגמה) להפעיל גם הם את המתפלים שלהם לאותו אירוע. שימוש לב שלו stopPropagation הינו מקבלים כאן את הודעה Hi ואת Hello.

מדריך אירועים - בניית סדר האירועים

עתה אתה יודע מהם אירועים וכייד הדפדף מטפל בהם. ראיית כיצד לכתוב פקודות המופעלות כאשר אירוע מתרחש, אבל עלייך לצלול קצת יותר פנימה לתוך העולם המופעל על ידי אירועים.

כפי שכבר למדת, פעולות משתמש יחידה או אירוע הקשור במערכת, מסוגלת להפעיל סדרת אירועים. דבר זה יכול להיעיל מפני שתוכל להשתמש באירועים שונים אלה כדי לטפל במצבים שונים, כמו באירועים onclick | onmousedown | onmouseup | onmousemove | onmouseout | onmouseover | onfocus | onblur | onclick | ondblclick ועודם בסופו של דבר גם גם צד שלילי. אם תכתוב קוד ייחיד למספר אירועים אפשריים, קוד זה עשוי לפעול בצורה שלא תרצה שיפעל. במקרה הגורע, רצף אירועים - כל אירוע והשיגרה שלו - עשויים לגרום למערכת להיכנס לולאה אינסופית. הצעדים הבאים מוכיחים ניקיטת צעדים מיוחדים למניע בעיות אלה:

- זיהוי האפשרות שמספר אירועים יתרחשו.
- זיהוי איזה אירועי יתרחשו כאשר המשתמש מבצע פעולה מסוימת.
- בניית סדר התראחותם האירועים.
- בדיקת הקשר בין אירועים שונים העולמים לפעול בו זמן. במיוחד יש לשים לב לאירועים onclick | onblur | onfocus | onmouseout | onmouseover | onmousedown | onmouseup | onclick | ondblclick לtaggit <a>

<a> במסגרת התגית onclick האירוע

בדפים קודמים נעשה שימוש באירועים onclick | onmouseout | onmouseover - במסגרת התגית. <a> כאשר נלחץ העבר על האובייקט (על הקישור) נתען לחalon הדפן הדף שכותבתו נמצא במאפיין href מה יקרה במצב בו תרצה לכתוב קוד שיופיע בעת התראחותם האירוע onclick לtaggit <a>

התיכון בקוד הבא:

1.
2. onclick="document.bgColor='Orange';">

מה לדעתך יקרה?

1. הרקע של המסמך ישנה לכתום ורק אחר כך יטען אתר הוד-עמי שכותבתו רשומה שמאפיין href.
2. הרקע של המסמך לא ישנה לכתום, יטען אתר הוד-עמי שכותבתו רשומה שמאפיין href.
3. הרקע של המסמך ישנה לכתום ואיתר הוד-עמי שכותבתו רשומה שמאפיין href לא יטען.
4. לא יקרה דבר בעת לחיצה על הקישור.

מכיון שאתה עשו שלא להבחן בהתרחשות אירועים בגל מהירות הוסף עוד משפט JavaScript אחד שיעזר לך לפניו - משפט alert():

1.
2. onclick="document.bgColor='Orange';alert('כתום');">

משפט alert() יציג תיבת דיווח ובה המילה כתום? שייהי עלייך לאשר. כך, תוכל לדעת בוודאות שאכן משפט JavaScript שנכתבו במסגרת אירוע onclick של התגית <a> אכן הופעל, אם הופעל. עכשו נסה לענות על השאלה.

ריבוי אירועים בפעולה יחידה

אחד הביעות בטיפול בריבוי אירועים היא שהמשתמש יוכל לתקשר עם התוכנית בהרבה צורות. לדוגמה, הוא יכול להציג לטיבת הטקסט בעזרת העבר או בעזרת מקש TAB היכן היה focus - לפני התנועה לティבה / או לפני הלחיצה על לחץ שלח טופס? כפי שתראה, ריבוי האפשרויות עלול לגרום לבלבול.

לעומת זאת, החידשות הטובות הן שלא כתוב קוד לרוב צירופי האירועים/ אובייקטים. לכן, למרות העובדה שהאירועים הנזכרים עלולים לkerot, התוכנית שתכתב תתעלם מהם ולפיכך הם לא יצרו שום בעיה.

למרות העובדה שאנו כתיבת הקוד המיותר הופכת את מלאכת הטיפול בربבי אירועים פשוטה יותר, ציריך עדין לבצע עבודה מסוימת. כדי לטפל במצב זה, הבנה ונסתכל על כמה סדרות אירועים אשר קוראות כאשר משתמש מבצע פעולה.

כך בדוגמא, לדוגמה, את תיבת הדוא-שייך המועדת לכינוס המשמש ל-Windows. אם נתאר לנו שתיבה זו נכתבת כדף HTML אז שבזמן הקלדת שם המשתמש והסימלה קוראים כמה אירועים.

קודם כל, הבנה נבחון הקשת מקש בודדת. לכל>To שנקליד בשם המשתמש מתרחשים שלושה אירועים: onkeypress, onkeydown ו-onkeyup (בסדר זה). כאשר אתה מזיז את הסמן משדה שם המשתמש אל שדה סיסמה, אתה מעביר את ה-focus מאובייקט אחד לשני, כך שתרחשים שני אירועים `zera`: עברו האובייקט הנוכחי (שם משתמש), ו-`on-focus` העבר האובייקט החדש (סיסמה).

לבסוף, כאשר לוחצים על לחץ OK, לחיצת העכבר פשוטה יוצרת שלושה אירועים `onmousedown`, `onmouseup` ו-`onclick`. I: `onmousedown`, `onmouseup` ו-`onclick`: `ondblclick` לאחר האירוע ו-`onfocus`, אם נלחץ לחיצה כפולה על אובייקט אחר, כגון הטופס, יווצרו שני אירועים נוספים לאחר העבר האירוע `onmouseup` נוספת. סך הכל יש כבר חמישה אירועים הקשורים למקרה שנראתה כפעולה יחידה של המשתמש. בנוסף, אם לחיצת העכבר גורמת ל-focus-על עברו מאובייקט אחד לשני, מתרחשים גם אירועים `blur` ו-`on-focus` הכל שבעה אירועים.

העובדת שניתן לבצע פעולות שונות כדי להגיע לאותה מטרה מהוות בעיה נוספת. לדוגמה, בואו נבחן את הלחץ OK בתיבת זיהוי המשתמש של Windows. האם ידעת שניתן להפעיל אותו על ידי לחיצה על העכבר וגם על ידי מעבר אליו עם מקש Tab והקשה על מקש הרוח? השימוש בעכבר לביצוע המשימה מפעיל את שגרות `onmousedown`, `onmouseup` ו-`on-click`, ולאחר מכן בשימוש במקלדת מפעיל את שגרות `onkeydown`, `onkeyup`, `onkeypress`, `onclick`. תחוליר מבלבל, הלא כן?

ואולם, ניתן לדעת איזה אירוע יקרו בזמן ריצת התוכנית ובאיזה סדר הם יקרו. באירוע של לחיצת העכבר הסדר הוא כזה:

`onmousedown`

`onmouseup`

`onclick`

במקרה של לחיצה כפולה הסדר הוא:

`onmousedown`

`onmouseup`

`onclick`

`ondblclick`

שים לב שאירוע `onclick` מתרחש רק לאחר שלחצן העכבר נלחץ (`onmousedown`) וושוחרר (`onmouseup`).

אירועים ב-JavaScript

אירועים המופעלים עבור אובייקט החלון. חל לtags:

תגובה	תיאור
ononafterprint	מפעיל סקריפט לאחר שהמסמך מודפס
onbeforeprint	מפעיל סקריפט לפני שהמסמך מודפס
	מפעיל סקריפט לפני טעינת מסמך
onblur	מפעיל סקריפט כאשר החלון מאבד פוקוס
onerror	מפעיל סקריפט כאשר שגיאה מתרחשת
onfocus	מפעיל סקריפט כאשר החלון מקבל פוקוס
onnonhaschange	מפעיל סקריפט כאשר המסמך השתנה
onload	מפעיל סקריפט כאשר המסמך טוען
onnonmessage	מפעיל סקריפט כאשר הודעה מופעלת
onnonoffline	מפעיל סקריפט כאשר המסמך עבר למצב offline
ononline	מפעיל סקריפט כאשר מסמך מגיע לstate online
onnonpagehide	מפעיל סקריפט כאשר החלון מוסתר
onnonpageshow	מפעיל סקריפט כאשר החלון הופך לגלוי
onnonpopstate	מפעיל סקריפט כאשר ההיסטוריה של החלון השתנה
onnonredo	מפעיל סקריפט כאשר המסמך מתבצע שוב
onnonresize	מפעיל סקריפט כאשר גודל החלון השתנה
onnonstorage	מפעיל סקריפט כאשר מסמך נתען
onnonundo	מפעיל סקריפט כאשר מסמך מבצע ביטול
onnonunload	סקריפט מופעל כאשר המשמש עוזב את המסמך

אירועים של טופס

אירועים מופעלים על ידי פעולות בתוך טופס HTML על כל האלמנטים, HTML אבל נפוץ יותר עבור אלמנטים של טופס:

תגובה	תיאור
onblur	מפעיל סקריפט כאשר אלמנט מאבד פוקוס
onchange	סקריפט מופעל כאשר אלמנט השתנה
onnoncontextmenu	סקריפט מופעל כאשר תפריט Kontextual מופעל
onfocus	מפעיל סקריפט כאשר אלמנט מקבל פוקוס
onnonformchange	מפעיל סקריפט כאשר טופס השתנה
onnonforminput	מפעיל סקריפט כאשר טופס מקבל קלט משתמש
onnoninput	מפעיל סקריפט כאשר אלמנט מקבל קלט משתמש
onnoninvalid	מפעיל סקריפט כאשר אלמנט לא חוקי
onreset	מפעיל סקריפט כאשר אלמנט מוחזר בHTML
onselect	מפעיל סקריפט כאשר אלמנט נבחר
onsubmit	מפעיל סקריפט כאשר טופס נשלח

תגובה	תיאור
onkeydown	מפעיל סקריפט כאשר מקש נלחץ
onkeypress	מפעיל סקריפט כאשר מקש נלחץ וושחרר
onkeyup	מפעיל סקריפט כאשר מקש שוחרר

אירוע עכבר

אירועים מופעלים על ידי עכבר, או פעולות דומות של משתמש. כל על כל האלמנטים HTML5.

תגובה	תיאור
onclick	סקרייפט מופעל בלחיצת עכבר
ondblclick	סקרייפט מופעל בלחיצה כפולה של עכבר
ondrag	סקרייפט מופעל בגיררת אלמנט
ondragend	סקרייפט מופעל בסיום גיררת אלמנט
ondragenter	סקרייפט מופעל כאשר אלמנט נגמר ליד חוקי
ondragleave	סקרייפט מופעל כאשר אלמנט עוזב יעד חוקי
ondragover	סקרייפט מופעל כאשר אלמנט עובר מעל יעד חוקי
ondragstart	סקרייפט מופעל בתחילה פעולה גירירה
ondrop	סקרייפט מופעל כאשר אלמנט הנגרר שוחרר
onmousedown	סקרייפט מופעל כאשר לחץ העכבר נלחץ
onmousemove	סקרייפט מופעל כאשר מצביע העכבר זו
onmouseout	סקרייפט מופעל כאשר מצביע העכבר יצא מאלמנט
onmouseover	סקרייפט מופעל כאשר מצביע העכבר עובר מעל האלמנט
onmouseup	סקרייפט מופעל כאשר לחץ העכבר שוחרר
onwheel	סקרייפט מופעל כאשר גלגל העכבר מסתובב
onscroll	סקרייפט מופעל כאשר גלילה של אלמנט מוגלה

Example

Set the background color of all elements in the document with class="example":

```
var x = document.querySelectorAll(".example");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "red";
}
```

Example

Set the background color of all <p> elements in the document:

```
var x = document.querySelectorAll("p");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "red";
}
```

Example

Set the border of all <a> elements in the document that have a "target" attribute:

```
var x = document.querySelectorAll("a[target]");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.border = "10px solid red";
}
```

Example

Set the background color of every <p> element where the parent is a <div> element:

```
var x = document.querySelectorAll("div > p");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "red";
}
```

Example

Set the background color of all <h2>, <div> and elements in the document:

```
var x = document.querySelectorAll("h2, div, span");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "red";
}
```

Parameter Values

Parameter	Type	Description
<i>CSS selectors</i>	String	Required. Specifies one or more CSS selectors to match. Selects HTML elements based on their id, classes, type, etc.

For multiple selectors, separate each selector with a comma (,).

Tip: For a list of all CSS Selectors, look at our [CSS Selector Reference](#).

Technical Details

DOM Version:	Selectors Level 1 Document Object
Return Value:	A NodeList object, representing all elements in the document that match the selector(s). The NodeList is a static collection, meaning that changes in the DOM will not affect it. It will throw a SYNTAX_ERR exception if the selector(s) is invalid.

More Examples

Example

Get all `<p>` elements in the document, and set the background color of the first `<p>` element (index 0):

```
// Get all <p> elements in the document
var x = document.querySelectorAll("p");

// Set the background color of the first <p> element
x[0].style.backgroundColor = "red";
```

Example

Get all `<p>` elements in the document with `class="example"`, and set the background of the first `<p>` element:

```
// Get all <p> elements in the document with class="example"
var x = document.querySelectorAll("p.example");

// Set the background color of the first <p> element with class="example" (index 0)
x[0].style.backgroundColor = "red";
```

Example

Find out how many elements with `class="example"` there are in the document (using the `length` property of the `NodeList` object):

```
var x = document.querySelectorAll(".example").length;
```

HTML DOM querySelectorAll() Method

Example

Get all elements in the document with class="example":

```
var x = document.querySelectorAll(".example");
```

More "Try it Yourself" examples below.

Definition and Usage

The querySelectorAll() method returns all elements in the document that matches a specified CSS selector(s), as a static NodeList object.

The NodeList object represents a collection of nodes. The nodes can be accessed by index numbers. The index starts at 0.

Tip: You can use the length property of the NodeList object to determine the number of elements that matches the specified selector, then you can loop through all elements and extract the info you want.

For more information about CSS Selectors, visit our [CSS Selectors Tutorial](#) and our [CSS Selectors Reference](#).

Browser Support

The numbers in the table specifies the first browser version that fully supports the method.

Method				
querySelectorAll()	4.0	9.0		3.5

Note: Internet Explorer 8 has support for CSS2 selectors. IE9 and later versions have support for CSS3 as well.

Syntax

```
document.querySelectorAll(CSS selectors)
```

More Examples

Example

Get the first `<p>` element in the document:

```
document.querySelector("p");
```

Example

Get the first `<p>` element in the document with `class="example"`:

```
document.querySelector("p.example");
```

Example

Change the text of an element with `id="demo"`:

```
document.querySelector("#demo").innerHTML = "Hello World!";
```

Example

Get the first `<p>` element in the document where the parent is a `<div>` element.

```
document.querySelector("div > p");
```

Example

Get the first `<a>` element in the document that has a "target" attribute:

```
document.querySelector("a[target]");
```

Example

This example demonstrates how multiple selectors work.

Assume that you have two elements: a `<h2>` and a `<h3>` element.

The following code will add a background color to the first `<h2>` element in the document:

```
<h2>A h2 element</h2>
<h3>A h3 element</h3>
```

```
document.querySelector("h2, h3").style.backgroundColor = "red";
```

However, if the `<h3>` element was placed before the `<h2>` element in the document. The `<h3>` element is the one that will get the red background color.

```
<h3>A h3 element</h3>
<h2>A h2 element</h2>
```

```
document.querySelector("h2, h3").style.backgroundColor = "red";
```

HTML DOM querySelector() Method

Example

Get the first element in the document with class="example":

```
document.querySelector(".example");
```

Definition and Usage

The querySelector() method returns the first element that matches a specified *CSS selector(s)* in the document.

Note: The querySelector() method only returns the first element that matches the specified selectors. To return all the matches, use the [querySelectorAll\(\)](#) method instead.

If the selector matches an ID in document that is used several times (Note that an "id" should be unique within a page and should not be used more than once), it returns the first matching element.

For more information about CSS Selectors, visit our [CSS Selectors Tutorial](#) and our [CSS Selectors Reference](#).

Browser Support

The numbers in the table specifies the first browser version that fully supports the method.

Method

querySelector()	4.0	8.0	3.5	3.2	10.0
-----------------	-----	-----	-----	-----	------

Syntax

```
document.querySelector(CSS selectors)
```

Parameter Values

Parameter	Type	Description
		Required. Specifies one or more CSS selectors to match the element. These are used to select HTML elements based on their id, classes, types, attributes, values of attributes, etc.
<i>CSS selectors</i>	String	For multiple selectors, separate each selector with a comma. The returned element depends on which element that is first found in the document (See "More Examples").

Tip: For a list of all CSS Selectors, look at our [CSS Selectors Reference](#).

Technical Details

DOM Version: Selectors Level 1 Document Object

Return Value: A NodeList object, representing the first element that matches the specified CSS selector(s). If no matches are found, null is returned. Throws a SYNTAX_ERR exception if the specified selector(s) is invalid.

Finding HTML Elements by Class Name

If you want to find all HTML elements with the same class name, use `getElementsByClassName()`.

This example returns a list of all elements with `class="intro"`.

Example

```
var x = document.getElementsByClassName("intro");
```



Finding elements by class name does not work in Internet Explorer 8 and earlier versions.

Finding HTML Elements by CSS Selectors

If you want to find all HTML elements that matches a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the `querySelectorAll()` method.

This example returns a list of all `<p>` elements with `class="intro"`.

Example

```
var x = document.querySelectorAll("p.intro");
```



The `querySelectorAll()` method does not work in Internet Explorer 8 and earlier versions.

Finding HTML Elements by HTML Object Collections

This example finds the form element with `id="frm1"`, in the forms collection, and displays all element values:

Example

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

JavaScript HTML DOM Elements

This page teaches you how to find and access HTML elements in an HTML page.

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors
- Finding HTML elements by HTML object collections

Finding HTML Element by Id

The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with id="intro":

Example

```
var myElement = document.getElementById("intro");
```

If the element is found, the method will return the element as an object (in myElement).

If the element is not found, myElement will contain null.

Finding HTML Elements by Tag Name

This example finds all <p> elements:

Example

```
var x = document.getElementsByTagName("p");
```

This example finds the element with id="main", and then finds all <p> elements inside "main":

Example

```
var x = document.getElementById("main");
var y = x.getElementsByTagName("p");
```

Finding HTML Objects

The first HTML DOM Level 1 (1998), defined 11 HTML objects, object collections, and properties. These are still valid in HTML5.

Later, in HTML DOM Level 3, more objects, collections, and properties were added.

Method	Description	DOM
document.anchors	Returns all <a> with a value in the name attribute	1
document.applets	Returns all <applet> elements (Deprecated in HTML5)	1
document.baseURI	Returns the absolute base URI of the document	3
document.body	Returns the <body> element	1
document.cookie	Returns the document's cookie	1
document.doctype	Returns the document's doctype	3
document.documentElement	Returns the <html> element	3
document.documentElementMode	Returns the mode used by the browser	3
document.documentElementURI	Returns the URI of the document	3
document.domain	Returns the domain name of the document server	1
document.domConfig	Returns the DOM configuration	3
document.embeds	Returns all <embed> elements	3
document.forms	Returns all <form> elements	1
document.head	Returns the <head> element	3
document.images	Returns all <image> elements	1
document.implementation	Returns the DOM implementation	3
document.inputEncoding	Returns the document's encoding (character set)	3
document.lastModified	Returns the date and time the document was updated	3
document.links	Returns all <area> and <a> elements value in href	1
document.readyState	Returns the (loading) status of the document	3
document.referrer	Returns the URI of the referrer (the linking document)	1
document.scripts	Returns all <script> elements	3
document.strictErrorChecking	Returns if error checking is enforced	3
document.title	Returns the <title> element	1
document.URL	Returns the complete URL of the document	1

JavaScript HTML DOM Document

With the HTML DOM, the document object is your web page.

The HTML DOM Document

In the HTML DOM object model, the document object represents your web page.

The document object is the owner of all other objects in your web page.

If you want to access objects in an HTML page, you always start with accessing the document object.

Below are some examples of how you can use the document object to access and manipulate HTML.

The next chapters demonstrate the methods.

Finding HTML Elements

Method	Description
<code>document.getElementById()</code>	Find an element by element id
<code>document.getElementsByTagName()</code>	Find elements by tag name
<code>document.getElementsByClassName()</code>	Find elements by class name

Changing HTML Elements

Method	Description
<code>element.innerHTML=</code>	Change the inner HTML of an element
<code>element.attribute=</code>	Change the attribute of an HTML element
<code>element.setAttribute(attribute,value)</code>	Change the attribute of an HTML element
<code>element.style.property=</code>	Change the style of an HTML element

Adding and Deleting Elements

Method	Description
<code>document.createElement()</code>	Create an HTML element
<code>document.removeChild()</code>	Remove an HTML element
<code>document.appendChild()</code>	Add an HTML element
<code>document.replaceChild()</code>	Replace an HTML element
<code>document.write(text)</code>	Write into the HTML output stream

Adding Events Handlers

Method	Description
<code>document.getElementById(id).onclick=function(){code}</code>	Adding event handler code to an onclick event

Example

The following example changes the content (the innerHTML) of the <p> element with id="demo":

```
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

In the example above, getElementById is a **method**, while innerHTML is a **property**.

The getElementById Method

The most common way to access an HTML element is to use the id of the element.

In the example above the getElementById method used id="demo" to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the **innerHTML** property.

The innerHTML property is useful for getting or replacing the content of HTML elements.



The innerHTML property can be used to get or change any HTML element, including <html>

What is the DOM?

The DOM is a W3C (World Wide Web Consortium) standard.

The DOM defines a standard for accessing documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

What is the HTML DOM?

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

JavaScript - HTML DOM Methods

HTML DOM methods are **actions** you can perform (on HTML Elements).

HTML DOM properties are **values** (of HTML Elements) that you can set or change.

The DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A **property** is a value that you can get or set (like changing the content of an HTML element).

A **method** is an action you can do (like add or deleting an HTML element).

JavaScript HTML DOM

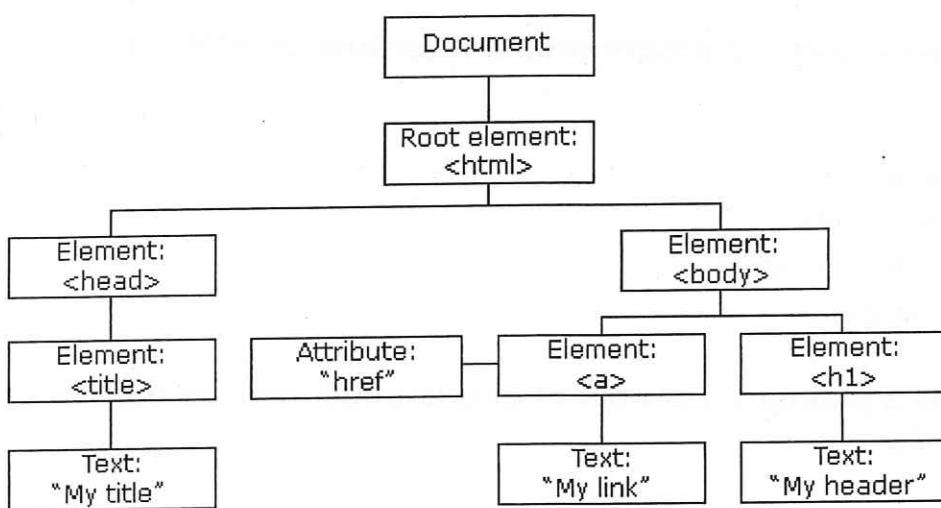
With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **Document Object Model** of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What You Will Learn

In the next chapters of this tutorial you will learn:

- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add and delete HTML elements

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8" />
5     <title></title>
6     <script type="text/javascript">
7         function func(event) {
8             document.getElementById('txt').value = event.target.value;
9             document.getElementById('txt2').value = event.currentTarget.id;
10        }
11
12
13     function f()
14     {
15         document.getElementById('txt3').value = 1 + 2;
16         document.getElementById('txt3').value = "1+2";
17         document.getElementById('txt3').value = eval("2^2");
18     }
19     </script>
20 </head>
21 <body>
22 <div id="div1" onclick="func(event)">
23     <input type="button" value="b1" id="b1" />
24     <input type="button" value="b2" id="b2" />
25 </div>
26     <input type="text" id="txt" />
27     <input type="text" id="txt2" />
28     <br />
29     <input type="text" id="txt3" />
30     <br /><input type="button" onclick="f()" />
31 </body>
32 </html>
```

```
1
2
3
4 <html dir="rtl" style="direction:rtl" >
5 <head>
6     <title>אימוחות נתונים טופס</title>
7
8
9     <style type="text/css">
10    label
11    {
12        float:right;
13        width:110px;
14        margin:10px;
15    }
16    input
17    {
18        padding:3px;
19        margin:10px;
20        width:200px;
21    }
22 </style>
23
24
25 <script type="text/javascript">
26     function cancelValidate() {
27         document.getElementById('frm1').noValidate = true;
28     }
29
30     function validate() {
31         if (document.getElementById('fname').value == "") {
32             alert("(! נא להכנס שם פרטי");
33             document.getElementById('fname').focus();
34             return false;
35         }
36
37         if (document.getElementById('lname').value == "") {
38             alert("(! נא להכנס שם משפחה");
39             document.getElementById('lname').focus();
40             return false;
41         }
42
43         if (document.getElementById('address').value == "") {
44             alert("(! נא להכנס כתובת");
45             document.getElementById('address').focus();
46             return false;
47         }
48         return true;
49     }
50
51     function checkTel() {
52         var txt = document.getElementById('tel').value;
53         for (var i = 0; i < txt.length; i++)
54             if(isNaN(txt[i])){
55                 //if (isNaN(txt.charAt(i))) {
56                 document.getElementById('tel').value = "";
```

```
57             alert("לא ניתן לחייב שמיון מס' טלפון");
58             document.getElementById('tel').focus();
59             return;
60         }
61     }
62
63     function checkChar(e) {
64         var charCode = (e.which) ? e.which : e.keyCode
65         if (charCode < 48 || charCode > 57)
66             return false;
67         return true;
68     }
69
70
71 </script>
72 </head>
73
74
75
76
77
78 <body onload="cancelValidate()">
79     <form name='frm1' id="frm1" onsubmit="return validate()" >
80         <div>
81             <div>
82                 <label><span style="color:Red">*</span> שם פרטי:</label>
83                 <input id="fname" name="fname" value="" size="20" type="text" />
84             </div>
85             <div>
86                 <label><span style="color:Red">*</span> שם משפחה:</label>
87                 <input id="lname" name="lname" required="required" size="20"
88                     type="text" />
89             </div>
90             <div>
91                 <label><span style="color:Red">*</span> כתובת:</label>
92                 <input id="address" name="address" required="required" size="25"
93                     type="text" />
94             </div>
95             <div>
96                 <label>עיר:</label>
97                 <input name="city" size="20" type="text" />
98             </div>
99             <div>
100                <label>טלפון:</label>
101                <input id='tel' name="tel" size="20" type="text"
102                    onchange="checkTel()"/>
103            </div>
104            <div>
105                <label>מיקוד:</label>
106                <input name="zip" size="20" type="text" onkeypress="return
107                    checkChar(event)" />
108            </div>
109        <div>
110            <label>Email:</label>
111            <input name="email" size="20" type="email" />
112        </div>
```

```
109      <div>
110          <input type="submit" name="submit" value="Submit" />
111          <input type="reset" value="Reset" />
112      </div>
113      <div>
114          <span style="color:Red">* דוחה קודש</span>
115      </div>
116  </div>
117 </form>
118 </body>
119 </html>
120
121
```

גְּדוֹלָה מִתְּבָרֶן תַּחַת כְּלֵלָה
בְּמִזְרָחָה - יְהוָה בְּרוּךְ הוּא

... שמייניגי ור שטינצ'ר \Events\אירועים\Copy of addEventListener.html

1

```
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4     <title>JavaScript HTML DOM EventListener</title>
5     <script>
6
7
8
9     //1b
10    function changeCss() {
11        document.getElementById('demo').style.backgroundColor = 'yellow';
12    }
13    //1
14    function displayDate() {
15        document.getElementById("demo").innerHTML = Date();
16        document.getElementById('demo').onmouseover = changeCss;
17    }
18
19    function start() {
20        /*2*/ document.getElementById("btn2").onclick = displayDate;
21
22        /*4*/ document.getElementById('div3').addEventListener("click",
23            funcdiv3);
24
25        /*5*/
26        document.getElementById('img1').addEventListener("mouseover",
27            listenerToimg);
28        document.getElementById('img1').addEventListener("mouseover",
29            listenerBToimg);
30
31        //3a
32        function func1a()
33        {
34            alert('onclick HTML markup');
35        }
36        //3b
37        function func1b() {
38            document.getElementById('btn1a').onclick = func1c;
39        }
40        //3c
41        function func1c() {
42            console.log('assign onclick using js');
43        }
44
45        function changeCss2()
46        {
47            document.getElementById('btn1a').style.backgroundColor = "pink";
48        }
49        function addEventfunc1() {
50            console.log("from addEvent");
51        }
52        function addEvent() {
53            document.getElementById('btn1a').addEventListener('click',
54                addEventfunc1);
55        }
56    }
57
```

```
53     document.getElementById('btn1a').addEventListener('mouseover', changeCss2);
54 }
55
56 //4a
57 function funcdiv3() {
58     console.log("click from div3");
59 }
60 //4b
61 function addToDiv3() {
62     if(!document.getElementById('btn3')) - btn3 הינו מוכן
63         document.getElementById('div3').innerHTML+="  
";
64 }
65
66 //5a
67 function listenerBToimg(e) {
68     e.target.style.border = "solid 5px yellow";
69 } הו
70
71 //5b
72 function listenerToimg()
73 {
74     console.log('mouseover the img1');
75 }
76
77 //5c פונקציית הולצת עזיבת המouserOVER
78 function removeMouseover() {
79     document.getElementById('img1').removeEventListener("mouseover", listenerToimg);
80 }
81
82 function changeCss3(id,color) {
83     document.getElementById(id).style.backgroundColor =color;
84 }
85
86 //6
87 function addEventanonymousfunc() { change על שרטון עזרתית וולקן
88     document.getElementById('txt1').addEventListener("change",
89         function () {
90             console.log("from anonymousfunc");
91             // this.style.backgroundColor = "pink";
92             // event.target.style.backgroundColor = 'yellow';
93         }
94
95         function addEventanonymousfunc2() {
96             document.getElementById('txt1').addEventListener("change",
97                 function () {
98                     changeCss3('txt1', "olive");
99                 });
100    }
101    function addEventanonymousfunc3() {
102        document.getElementById('txt1').addEventListener("click", function f() {
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>HTML DOM classList Property</title>
5
6     <style>
7         .mystyle {
8             width: 300px;
9             height: 50px;
10            background-color: coral;
11            color: white;
12            font-size: 25px;
13        }
14        .mystyle2 {
15            text-decoration:underline;
16            outline:2px solid green;
17        }
18    </style>
19
20    <script>
21        function myFunction() {
22            document.getElementById("myDIV").classList.add
23                ("mystyle", "mystyle2");
24
25            // document.getElementById("myDIV").classList = "mystyle";
26            // document.getElementById("myDIV").classList += " mystyle2";
27
28            // document.getElementById("myDIV").setAttribute("class", "mystyle
29                mystyle2");
30        }
31
32        function myFunction2() {
33            document.getElementById("myDIV").classList.remove("mystyle");
34        }
35
36        function myFunction3()
37        {
38            document.getElementById("myDIV").classList.toggle("mystyle2",true);
39        }
40
41        function myFunction4() {
42            console.log( document.getElementById("myDIV").classList.contains
43                ("mystyle2"));
44        }
45
46    </script>
47
48 <body>
49 <p>Click the button to add the "mystyle" class to DIV.</p>
50
51 <button onclick="myFunction()">add class</button><br />
52 <button onclick="myFunction2()">remove class</button><br />
53 <button onclick="myFunction3()">toggle class</button><br />
```

```
54 <button onclick="myFunction4()">contains class</button><br />
55
56
57 <p><strong>Note:</strong> The classList property is not supported in Internet Explorer 9 and earlier versions.</p>
58
59 <div id="myDIV">
60 I am a DIV element
61 </div>
62
63 </body>
64 </html>
65
```



```
49         str = str + " שאלת הגיל: " + document.getElementById("selectAge").options[selectItem].value;
50         alert(str);
51     }
52
53
54     function addNamesInSelect() {
55         if (!(document.getElementById('addNames'))){  
56             document.getElementById('div1').style.display = 'block';
57             document.getElementById('div1').innerHTML = "<select  
name='addNames' id='addNames' size='8'></select>";
58         }
59         var i = document.getElementById('selectNames').selectedIndex;
60         var name = document.getElementById('selectNames').options[i].text;
61         document.getElementById('addNames').options  
[document.getElementById('addNames').length] = new Option(name);
62     }
63 </script>
64 </head>
65
66
67
68
69
70
71
72
73
74
75
76
77 <body>
78     <form id='frm1' name="frm1">
79
80         סיסמה:<input type="password" id="pass" name="pass" size="5"  
maxLength="5" onBlur="checkPassword();"/><p id="msg"  
onchange="ff()"></p>
81         <br /><br />
82         <input type="checkbox" name="chkCredit"  
onClick="setText('chkCredit', 'txtCredit', 'כרטיס אשראי');"/>  
סמן אשראי
83
84         <input type="text" name="txtCredit" size="10" maxLength="20"  
disabled="disabled" />
85         <input type="checkbox" name="chkDelivery"  
onClick="setText('chkDelivery', 'txtchkDelivery', 'පוילוש');"/>  
סמן משלוח
86
87         <input type="text" name="txtchkDelivery" size="10" maxLength="20"  
disabled="disabled" />
88
89         <hr style="color:Blue" />
90         <input type="radio" name="radioBgcColor" value="yellow"  
onClick="changeBgColor(0)"/>צהוב<br />
91         <input type="radio" name="radioBgcColor" value="red"  
onClick="changeBgColor(1)"/>אדום<br />
92         <input type="radio" name="radioBgcColor" value="pink"  
onClick="changeBgColor(2)"/>розי<br />
```

```
93      <input type="button" value="הציג הודעה - מספר הצעב הנבחר" onclick="alertColor()" />
94
95      <br /><br />
96      <select id='selectAge' name="selectAge">
97          <option value="-10">10 ו מתחת</option>
98          <option value="10-13">10-13 <input checked="" type="radio"/>
99          <option value="14-17" selected="selected">14-17 </option>
100         <option value="18-21" >18-21 </option>
101         <option value="22-25" >22-25 </option>
102         <option value="26-29" >26-29 </option>
103         <option value="30-33" >30-33 </option>
104         <option value="34-37" >34-37 </option>
105         <option value="+37" >37 ומעלה</option>
106     </select>
107
108     <input type="button" value="פרטינט" onclick="showSelect()" />
109
110     <div id="div1" style="display:none">
111     </div>
112
113     <br/><br/> להוספה נוספת כפולה<br/>
114     <select id="selectNames" name="selectNames" size="8" ondblclick="addNamesInSelect()">
115         <option>רשות</option>
116         <option>רבקה</option>
117         <option>ליז</option>
118         <option>ליאה</option>
119         <option>תרי</option>
120         <option>זהב</option>
121     </select>
122
123     </form>
124 </body>
125 </html>
126
127
```

```
1 <!DOCTYPE html>
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4     <title>JavaScript HTML DOM Elements</title>
5     <style type="text/css">
6         .cls1 {
7             }
8         .cls2 {
9             }
10        p::first-line {
11            font-size:20px;
12        }
13    </style>
14    <script type="text/javascript">
15
16        function func() {
17
18            dir ↴ מודרנ
19            document.getElementById('p1').setAttribute('dir', 'rtl');
20            document.getElementById('p1').removeAttribute('dir');
21            document.getElementById('p1').style.textDecoration = "underline";
22
23            var x;
24            x = document.getElementById('p1');
25            x = document.getElementById('ttttttttt');
26
27            x = document.getElementsByClassName('cls1');
28            x = document.getElementsByClassName('tttttt');
29
30            x = document.getElementsByName('r1');
31
32            //      document.getElementsByClassName('cls2').style.Color =
33            //      "red"; //error ↪ תקע בפונקציית getElementsByClassName()
34            document.getElementsByClassName('cls2')[1].style.color = "red"; - error
35            ↪
36            x = document.getElementsByTagName('p');
37            for (i = 0; i < x.length; i++)
38                x[i].style.backgroundColor = "orange";
39
40
41            querySelectorAll ↪ פונקציית שולחן
42            var ll = document.querySelector(".cls1"); ↪ מילוי של גירסאות ישנות
43            li = document.querySelector(".trtrtr"); ↪ מילוי של גירסאות ישנות
44            ↪
45            document.querySelector(".cls1").style.border = "2px solid blue";
46            ↪ id/class מופיע בזיהוי
47            var x = document.querySelectorAll(".cls1 , .cls2"); - ↪ מילוי של גירסאות ישנות
48            for(i=0;i<x.length;i++)
49                x[i].style.outline = "2px solid blue";
50            x = document.querySelectorAll(".tttttttt");
51
52            document.getElementById('p1').setAttribute('dir', 'rtl');
53            document.querySelector("p[dir]").style.color = "pink";
54
55            //var yy= document.querySelector("p::first-line");//null
```

innerHTML
innerHTML - (בנ"ה וע)

...ment querySelector_All dom\Copy of querySelector.html

```
56     //yy = document.querySelectorAll("p::first-line");//ר'ג ר'ג
57
58     yy = document.querySelectorAll("p:first-child");//yy = [p#p1.cls2, ↵
      p, p]
59     yy = document.querySelectorAll("p:nth-child(2)");//yy = [p, p]
60
61
62     var zz = document.querySelector('#div1');
63     var xx = zz.querySelector('div p');
64     console.log(xx.innerText);
65     console.log(document.getElementById('div1').querySelector('
      p').innerText);
66
67
68     var listLi = document.getElementById('ul1').querySelectorAll
      ('li');
69     console.log(listLi.length);
70     document.getElementById('ul1').innerHTML += "<li>ddddddd</li>";
71     console.log(listLi.length);
72     listLi[0].innerHTML = "first changed";
73     console.log(listLi[0].innerHTML);
74
75
76     var listLi2=document.getElementById('ul1').getElementsByName
      ('li');
77     console.log(listLi2.length);
78     document.getElementById('ul1').innerHTML += "<li>ddddddd</li>";
79     console.log(listLi2.length);
80     listLi2[1].innerHTML = "changed";
81
82     var pHtmlCollection = document.getElementsByName('p');
83     var pnodeList = document.querySelectorAll('p');
84     ↪'נ'ל פ'ר
85     var docu = document.links;
86     console.log(docu.length);
87     console.log(docu[0].innerHTML);
88     docu[0].innerHTML = 'ttttt';
89     var x = document.getElementsByName('a');
90     x[1].innerHTML = 'bbbbbbb';
91     console.log(docu[1].innerHTML);
92   }
93 </script>
94 </head>
95 <body onload="func()">
96 <p id="p1" class="cls2" >set/removeAttribute</p>
97   <p>hjkjhjhj<a id="a1" href="dsfasddfas">link</a>kh</p>
98 <p id="p11" class="cls2">cls2 b</p>
99 <p id="p2" class="cls1">jkdflsdkjfklsd</p>
100 <p class="cls1" dir="rtl">jkdflsdkjfklsd</p>
101 <p class="cls1">jkdflsdkjfklsd</p>
102 <p class="cls1">jkdflsdkjfklsd</p>
103 <p class="cls1">jkdflsdkjfklsd<br />dafsafadsf<br />dafsfdasaf<br />dfsafddas</
      p>
104 <!-- <input name="p2" type="text" size="20" /> -->
105   <div id="div1">
106     <p>p in div1</p>
```

```
107      <p>p in div1 B</p>
108      <ul id="ul1"><li>aaaa</li>
109          <li>bbbbbb</li>
110          <li>cccccc</li>
111      </ul>
112  </div>
113  <div id="div2">
114      <p>p in div2</p>
115  </div>
116      <a href="sdfd">dsadf</a><a href="dsfa">link3</a>
117  <form>
118      <input type="radio" name="r1" />option 1
119      <input type="radio" name="r1" />option 2
120      <input type="radio" name="r1" />option 3
121  </form>
122 </body>
123 </html>
124
```

```
1 <!DOCTYPE html />
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4
5     <title>Event Bubbling or Event Capturing?</title>
6
7     <style type="text/css">
8         div
9         {
10             border: 2px solid black;
11         }
12         #div1
13         {
14             width:400px;
15             height:400px;
16             background-color:Orange;
17         }
18         #div2
19         {
20             width:300px;
21             height:300px;
22             position: relative;
23             top:10px;
24             left:10px;
25             background-color:Gray;
26         }
27         #div3
28         {
29             width:200px;
30             height:200px;
31             position: relative;
32             top:10px;
33             left:10px;
34             background-color:Fuchsia;
35         }
36     </style>
37
38     <script type="text/javascript">
39         //function funcOnLoad() {
40             //    document.getElementById('div1').addEventListener('click',
41             //        function (event) {
42                 //            document.getElementById("txt1").value=event.target.id;
43                 //            document.getElementById("txt2").value=event.currentTarget.id;
44                 //            if (event.target == this)
45                 //                if(event.target==event.currentTarget)
46                 //                    document.getElementById
47                 //                        (event.currentTarget.id).setAttribute('dir', 'rtl');
48             });
49             //    document.getElementById('bt1').addEventListener('click',
50             //        function (event) {
51                 //            document.getElementById('bt1').style.color = 'red';
52                 //            event.stopPropagation();
53                 //            event.stopImmediatePropagation();
54             });
55             //    document.getElementById('bt1').addEventListener('click',
56             //        function () {
57                 //            document.getElementById('bt1').style.backgroundColor =
58                 //                'yellow';
59             });
60     </script>
```

```
51     // document.getElementById('bt1').addEventListener('mouseleave', function () {  
52     //     document.getElementById('bt1').parentNode.style.backgroundColor = 'olive';  
53     // });  
54     // document.getElementById('div3').addEventListener('click', function (event) {  
55     //     alert("from div3 click");  
56     // });  
57     // document.getElementById('div2').addEventListener('click', function (event) {  
58     //     alert("from div2 click");  
59     // });  
60     // document.getElementById('bt3').addEventListener('click', function (event) {  
61     //     alert("from bt3 click");  
62     //     event.stopPropagation();  
63     // });  
64 }  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76     function funcOnLoad() {  
77         document.getElementById('div1').addEventListener('click', function (event) {  
78             console.log("from div1 click");  
79             document.getElementById("txt1").value = event.target.id;  
80             document.getElementById("txt2").value =  
81                 event.currentTarget.id;  
82             if (event.target == this)  
83                 document.getElementById((event.currentTarget.id).setAttribute('dir', 'rtl'));  
84             }, true);  
85             document.getElementById('bt1').addEventListener('click', function (event) {  
86                 document.getElementById('bt1').style.color = 'red';  
87             }, true);  
88             document.getElementById('bt1').addEventListener('click', function () {  
89                 document.getElementById('bt1').style.backgroundColor =  
90                     'yellow';  
91             }, true);  
92             document.getElementById('bt1').addEventListener('mouseleave', function (event) {  
93                 //     event.currentTarget.parentNode.style.backgroundColor = 'olive';  
94                 //     this.parentNode.style.backgroundColor= 'olive';
```

```
95    //      event.target.parentNode.style.backgroundColor = 'olive';
96    document.getElementById('bt1').parentNode.style.backgroundColor = 'olive';
97 }, true);
98 document.getElementById('div3').addEventListener('click', function (event) {
99     console.log("from div3 click");
100}, true);
101 document.getElementById('div2').addEventListener('click', function (event) {
102     console.log("from div2 click");
103}, true);
104 document.getElementById('bt3').addEventListener('click', function (event) {
105     console.log("from bt3 click");
106     event.stopPropagation();
107 }, true);
108 }
109
110 </script>
111 </head>
112 <body onload="funcOnLoad()">
113     <div id='div1'>
114         <p id='p1' style="border:solid 2px pink">paragraph 1 line 1<br />
115             >paragraph 1 line 2</p>
116         <div id='div2'>
117             <p id='p2'>paragraph 2 line 1<br />paragraph 2 line 2</p>
118             <div id='div3'>
119                 <label>event.target</label> <input type="text" id='txt1' size=20 /><br />
120                 <label>event.currentTarget</label> <input type="text" id='txt2' size=20 /><br />
121                 <input type="button" id="bt1" value="bt1" /><br />
122                 <input type="button" id="bt2" value="bt2" /><br />
123                 <input type="button" id="bt3" value="bt3" /><br />
124             </div>
125         </div>
126     </body>
127 </html>
```

מבנה פרויקט נכון בHTML

מדרך קצר זה מתרגם את המבנה הנכון של עמוד הנקטב בקוד HTML פג' ט כמה ניטים לצמצום הקוד, ומודגש כמה נקודות השובות בכתיבה קוד נקי וטוב.

כללים לכתיבה קוד HTML תקין:

- ✓ אל תקצרו את הקוד על חשבון תקינות הקוד - ככל שהקוד יהיה נכון וധוק, כך הוא יהיה טוב יותר. גולג מיחס חשיבות לתקינות הקוד ולזמן עליית העמוד. הזמן הנוכחי משמש יעילה מתקינות הקוד - אם הקוד נכון, הדפסן לא מתאים והוא נקבל את העמוד מהו יותר.
- ✓ תחילה בכתיבה HTML שלכם תחילת וرك לאחר מכן כתבו את הCSS. ברגע שככל אלמנטי HTML יהיו מוכנים, יהיה לכם קל יותר למפות את האתר ולהחליט על עיצוב של כל פרט בצורה מרכזת ונכונה.
- ✓ בצעו שימוש בתגים המבנימים החדשניים של HTML 5, כך תתנו ערך ספציפי לכל תגיה מה שיעזר לדירוג שלכם במניע החיפוש.
- ✓ יש לשמר על הזוחות נכונות, כך הקוד נעים יותר לעין וניתן לראות את זירוכית האלמנטים בצורה נכונה.
- ✓ תגי HTML רוחניים באoitיות קטנות <body> ולא <BODY>.
- ✓ מומלץ להוסיף Tag meta בHead של הדף, לצורך נתינת משמעות סתמית לדף. (ראו בדוגמה למטה)
- ✓ תנו שמות קבועים ותמונה באנגלית בלבד, יהיה לכם הרבה יותר קל ליבור עם זה אחרך.
- ✓ יש להקפיד על שמותמשמעותיים לקבצים ולחומרונות.
- ✓ אל תחסכו במאפיינים יאל תגים לצורך צמצום קוד, לרבות זה יותר פוגע במולש מאשר עוזר למגל.
- ✓ אל תכתבו קוד צפוח, גולג לא מגביל אתכם בכמות ה- Enter ים.
- ✓ עבור כל קישור יש לשים את המאפיין title שנותן משמעות סמנטית ל קישור עבור מיעי החיפוש.
- ✓ עבור כל תמונה חובה להוסיף מאפיין alt גם לצורך המשמעות הסהמונית והשימוש מקרים שבה לא יטענו התמונות (כגון כשהמשתמש חסם תמונה או עבר עוורים שמערכת מקראית להם את המשמעות התמונה) אלא רק alt.
- ✓ נסו לעצב את הטקסט בעזרת CSS ככל האפשר, זה מיצמצם הרבה.

- ✓ המנעו שימוש ב style פנימי, מקום של ה Styles בקובץ css.
- ✓ יש לבצע הפרדה בין קוד HTML לבין CSS או Javascript מכך שwon't ישם לא מעוניינים שקו ה CSS יופיע בעמוד שלו בלבד שוגול מתוך ה HTML - של העמוד, וגם מעוניינים שההגעה של העיבושים של גוגל לתוכה תהיה מהירה ככל האפשר.
- כמו כן הפרדה שומרת על קריאות הקוד, ומאפשרת שימוש מרוכז באותו קבצים עבור האתר כל, ללא הכפלה מיותרת.
- ✓ דוגמא למבנה דף נcano עם שימוש בתמונות המבניות החדשניות HTML5

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>
  <head>
    <meta name="keywords" content="השנה מילון" />
    <meta name="description" content="השנה מילון" />
    <title> Devschool.co.il </title>
  </head>

  <body>
    <header>
      <h1>Devschool.co.il</h1>
      <h2>Free online web development school</h2>
    </header>

    <nav>
      <ul>
        <li>Home</li>
        <li>HTML 5</li>
        <li>SQL</li>
      </ul>
    </nav>

    <section>
      <h1>My Article</h1>
      <article>
        <p>...</p>
      </article>
    </section>

    <footer>
      <p>...</p>
    </footer>
  </body>
</html>
```

כללים ל CSS נכון:

- + כאשר אתם כותבים חוקי CSS, חלקכם משתמשים בשורה אחת וחלקכם משתמשים במספר שורות כה:

```
.BigDiv {  
background: blue;  
border: 1px solid black;  
}
```

```
.BigDiv { background: blue; border: 1px solid black; }
```

שני חוקי CSS בדוגמה עושים אותו הדבר וזרת הכתיבה של שניהם נכון.

- + פה נכנס עניין הנוחות, בחרו את הצורה הנוחה ביותר לשיליכם לקודם אחר הצמדה אליה אל תעשו חצי מהקוד בצורה כזו וחצי בצורה אחרת - זה מקשה על קריית הקוד.

- + השתמשו בשמות רלוונטיים, המנעו שימוש בשמות לסלקטורים שלא ציידים על תפקוד הסלקטור (כמו Z,x,div1,moshe וכדומה).

- + השתמשו בהערות!
קוד עם העות יחסור לכם זמן כאשר תתחזקו את קודך CSS ויקל על הקראיה שלהם.
הערות צרכות להוות בחרות ולהuid על תפקוד חוק CSS.
דוגמא להערה בסיס:

```
*-- Big Div in the middle --*/
```

```
.BigDiv { background: blue; border: 1px solid black; }
```

- + מומלץ לחלק את קודך CSS לשורות (אזורים) לפי חלקי האתר - כך תוכלן לעבוד בצורה מסודרת.

דוגמא לחלוקת:

```
*---GENERIC---*/
```

```
/*---HEADER---*/
```

```
/*---CONTENT---*/
```

```
/*---MAIN MENU ---*/
```

/*---SIDE MENU---*/

/*---FOOTER---*/

- + CSS יש לכתב בצורה מדווגת מהכל אל הפרט, כמו למשל להגדיר תחילת דברים כלליים. פונקציה, צבע רקע ועוד, לאח"כ את ה header, לאחר מכן את התוכן, אז התפריטים וכך הלאה (כתבו מלמעלה למטה).
- + יש להעדיה שימוש בסלקטור של קלאס מאשר בסלקטור של פאץ' מכיוון שבאותו קלאס ניתן להשתמש עבור כמה אלמנטים וכך הקוד נקי יותר, וכל לתחזוקה.
- + אם יש לכם מקורה שבו יש לכם כמה סלקטורים העשויים אותו הדבר, נבצע אותם בסלקטור אחד, כך שכתצרכו לעשות שינוי, תעשו אותו במקום אחד ולא בשלושה.

לדוגמא:

```
.news {
    background: #eee;
    border: 1px solid #ccc;
    border-radius: 6px
}
.events {
    background: #eee;
    border: 1px solid #ccc;
    border-radius: 6px
}
.feat-box {
    background: #eee;
    border: 1px solid #ccc;
    border-radius: 6px;
}
```

ניתן לאחד כל

- + השתמשו בקיצורים (Shortend), קיצורים יקטינו משמעותית את גודל קובץ ה CSS שלהם. לדוגמה כאשר יש להגיד גם margin left וגם margin right השתמשו בקיצור margin.

מבנה פרויקט נכון - Javascript

1. השתמשו ב `==` בכל מקרה אפשרי. (`==` מבצע המרה – בכל מקרה בו ההשוואה היא בין משתנים מסוימים).
2. אל תשתמש בקיצורים: גם כיש רק פקודה אחת בתוך `if`, לדוגמה:

```
if(someVariableExists)
x = false
```

שיםו אותה בתוך סוגרים – זה משפר את קריאות הקוד.

```
if(someVariableExists){
x = false
}
```

3. את תגיית `script` יש לשים בסוף העמוד – המטרה העיקרית היא לגרום לעלות מהר ככל האפשר, שימושים `script`, הדפדף לא יכול להמשיך עד שכולו עולה, והמשתמש יאלץ לחכות יותר זמן, הטוב ביותר הוא למקם את תגיית `script` לפני הסגירה של `body`:

```
<script type="text/javascript" src="path/to/anotherFile.js"></script>
</body>
</html>
```

4. הקצתה משתנים תעשה מחוץ ללולאה

Bad

```
1 for(var i = 0; i < someArray.length; i++) {
2     var container = document.getElementById('container');
3     container.innerHTML += 'my number: ' + i;
4     console.log(i);
5 }
```

Better

```
1 var container = document.getElementById('container');
2 for(var i = 0, len = someArray.length; i < len; i++) {
3     container.innerHTML += 'my number: ' + i;
4     console.log(i);
5 }
```

5. הוסיפו הערות לקוד

הערות מוסיפים לקוד שלא ברור בקריאה ראשונית מה הוא עשה.

לפעמים זה נראה מיותר בתחילת, אבל בהמשך צריך את זה מאוד (הרבה פעמים משתמשים על פונקציות ישנות שכתבו בעבר).

פונקציות פשוטות לא צרכות תמיד, בתנאי שימושם בשמות שימושיים לפונקציה ולמשתנים.

6. מקומם של קבצי `css` הוא בתוך תגיית `head`.

7. להצהר שימוש מיותר באלמנטים – האינסטינקט הראשוני שלנו, הוא לעטוף כל פסקה בDIV ואז לעתוף אותה שוב ליתר ביטחון... צורת כתיבה זו היא חסרת ייעילות ומסרבלת.

8. תוכן העיקרי של העמוד מגע לאחר החלק העליון של העמוד והtransform העליוון, אבל לפני כל חלק נלווה אחר כמו sidebar

מבנה פרויקט

assets

- CSS
- JS
- lib
- img

index.html

JavaScript Forms

JavaScript Form Validation

HTML form validation can be done by a JavaScript.

If a form field (fname) is empty, this function alerts a message, and returns false, to prevent the form from being submitted:

JavaScript Example

```
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == null || x == "") {
        alert("Name must be filled out");
        return false;
    }
}
```

The function can be called when the form is submitted:

HTML Form Example

```
<form name="myForm" action="demo_form.asp" onsubmit="return
validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

HTML Form Validation

HTML form validation can be performed automatically by the browser:

If a form field (fname) is empty, the **required** attribute prevents this form from being submitted:

HTML Form Example

```
<form action="demo_form.asp" method="post">
<input type="text" name="fname" required>
<input type="submit" value="Submit">
</form>
```



HTML form validation does not work in Internet Explorer 9 or earlier.

Data Validation

Data validation is the process of ensuring that computer input is clean, correct, and useful.

Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct input to a computer application.

Validation can be defined by many different methods, and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.

Client side validation is performed by a web browser, before input is sent to a web server

HTML Constraint Validation

HTML5 introduced a new HTML validation concept called **constraint validation**.

HTML constraint validation is based on:

- Constraint validation **HTML Input Attributes**
- Constraint validation **CSS Pseudo Selectors**
- Constraint validation **DOM Properties and Methods**

Constraint Validation HTML Input Attributes

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

For a full list, go to [HTML Input Attributes](#).

Constraint Validation CSS Pseudo Selectors

Selector Description

:disabled	Selects input elements with the "disabled" attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no "required" attribute specified
:required	Selects input elements with the "required" attribute specified
:valid	Selects input elements with valid values

For a full list, go to [CSS Pseudo Classes](#).

JavaScript Validation API

Constraint Validation DOM Methods

Property	Description
checkValidity()	Returns true if an input element contains valid data.
setCustomValidity()	Sets the validationMessage property of an input element

If an input field contains invalid data, display a message:

The checkValidity() Method

```
<input id="id1" type="number" min="100" max="300">
<button onclick="myFunction()">OK</button>

<p id="demo"></p>

<script>
function myFunction() {
    var inpObj = document.getElementById("id1");
    if (inpObj.checkValidity() == false) {
        document.getElementById("demo").innerHTML = inpObj.validationMessage;
    }
}
</script>
```

Constraint Validation DOM Properties

Property	Description
validity	Contains boolean properties related to the validity of an input element.
validationMessage	Contains the message a browser will display when the validity is false.
willValidate	Indicates if an input element will be validated.

Validity Properties

The **validity property** of an input element contains a number of properties related to the validity of data:

Property	Description
customError	Set to true, if a custom validity message is set.

patternMismatch	Set to true, if an element's value does not match its pattern attribute.
rangeOverflow	Set to true, if an element's value is greater than its max attribute.
rangeUnderflow	Set to true, if an element's value is less than its min attribute.
stepMismatch	Set to true, if an element's value is invalid per its step attribute.
tooLong <i>खेल पर अल्प</i>	Set to true, if an element's value exceeds its maxLength attribute.
typeMismatch	Set to true, if an element's value is invalid per its type attribute.
valueMissing	Set to true, if an element (with a required attribute) has no value.
valid	Set to true, if an element's value is valid.

Examples

If the number in an input field is greater than 100 (the input's max attribute), display a message:

The rangeOverflow Property

```
<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>
<p id="demo"></p>

<script>
function myFunction() {
    var txt = "";
    if (document.getElementById("id1").validity.rangeOverflow) {
        txt = "Value too large";
    }
    document.getElementById("demo").innerHTML = txt;
}
</script>
```

If the number in an input field is less than 100 (the input's min attribute), display a message:

The rangeUnderflow Property

```
<input id="id1" type="number" min="100">
<button onclick="myFunction()">OK</button>
<p id="demo"></p>
<script>
function myFunction() {
    var txt = "";
    if (document.getElementById("id1").validity.rangeUnderflow) {
        txt = "Value too small";
    }
    document.getElementById("demo").innerHTML = txt;
}
</script>
```

כניסה לשירות online

קוד משתמש:

סיסמה:

שלח

קוד אישר ו/או סיסמה שגויה

כניסה לשירות online

קוד משתמש:

סיסמה:

שלח

111 aaa
222 bbb
333 cccc
444 ddd

111 aaa
222 bbb
333 cccc
444 ddd

כניסה לשירות online

קוד משתמש:

סיסמה:

שלח

לא ניתן לשולח במשך דקה

כניסה לשירות online

קוד משתמש:

סיסמה:

שלח

5 ניסיונות טגויים, לא ניתן לשולח נתונים במשך דקה

111 aaa
222 bbb
333 cccc
444 ddd

111 aaa
222 bbb
333 cccc
444 ddd

```

1 function myFunction() {
2     var para = document.createElement("P");
3     para.style.color = 'green';
4     para.id = 'p1'; נאנו ב
5     para.setAttribute("dir", "rtl");
6     var attr = document.createAttribute("title");
7     attr.value = "from createAttribute"; ב
8     para.setAttributeNode(attr);
9     var t = document.createTextNode("This is a paragraph.");
10    para.appendChild(t); נאנו מ
11    //para.innerHTML = t.nodeValue;;
12
13    document.body.appendChild(para);
14
15    console.log(document.getElementById('p1').innerHTML);
16    console.log(document.getElementById('p1').childNodes[0].nodeValue);
17    console.log(document.getElementById('p1').attributes);
18    console.log(document.getElementById('p1').attributes[3].value);
19
20    document.getElementById('p1').style.backgroundColor = "pink";
21    para.style.border = 'solid 2px olive';
22    נאנו ב
23    var c = document.createComment("appendChild comments");
24    document.body.appendChild(c);
25
26    para = document.createElement("p");
27    var node = document.createTextNode("This is new.");
28    para.appendChild(node); נאנו ב, נאנו ב, נאנו ב, נאנו ב, נאנו ב
29
30    document.getElementById("div1").insertBefore(para, document.getElementById("pa2"));
31
32    para = document.createElement("p");
33    para.setAttribute("id", "paEnd"); ← נאנו ב
34    para.appendChild(document.createTextNode("This is also new"));
35    document.getElementById("div1").insertBefore(para, null);
36 }
37
38 //var parent = document.getElementById("div1");
39 //var child = document.getElementById("p1");
40 //parent.removeChild(child);
41 document.getElementById('removeb').onclick = function () { נאנו ב
42     if (document.getElementById('pa1')) ↴ נאנו ב
43         document.getElementById('pa1').parentElement.removeChild
44             (document.getElementById('pa1'));
45
46 document.getElementById('replaceb').onclick = function () {
47     var para = document.createElement("p");
48     var node = document.createTextNode("from replace");
49     para.appendChild(node);
50     var parent = document.getElementById("div1");
51     var child = document.getElementById("pa2");
52     parent.replaceChild(para, child);
53
54     var p=document.getElementById("pa1");

```

```
55     document.getElementById('div1').replaceChild(document.createElementId  
      ("pa3"), p);  
56     document.body.appendChild(document.getElementById("pa3"));  
57 }  
58  
59 document.getElementById('after').onclick = function () {  
60     var param=document.createElement('p');  
61     param.appendChild(document.createTextNode("insert after"));  
62     var p = document.getElementById("pa2");  
63     document.getElementById("div1").insertBefore(param, p.nextElementSibling);  
64  
65     param = document.createElement('p');  
66     param.appendChild(document.createTextNode("again - insert after"));  
67     p = document.getElementById("paEnd");  
68     document.getElementById("div1").insertBefore(param, p.nextElementSibling);  
69 }  
70  
71 document.getElementById('move').onclick = function () {  
72     var node = document.getElementById("myList2").lastChild;  
73     var list = document.getElementById("myList1");  
74     list.insertBefore(node, list.childNodes[0]);  
75 }  
76  
77 document.getElementById('clone').onclick = function () {  
78     var para = document.createElement('p');  
79     para.innerHTML = "p in div2";  
80     document.getElementById('div1').appendChild(para);  
81  
82     document.getElementById('div2').appendChild(para);  
83  
84     var param = para.cloneNode(true);  
85     param.id = 'pnew';  
86  
87     document.getElementById('div2').appendChild(param);  
88     param.style.backgroundColor = 'gray';  
89  
90     var param2 = document.getElementById('p1').cloneNode(true);  
91     param2.id = 'pnew2';  
92     document.getElementById('div2').appendChild(param2);  
93  
94  
95  
96     document.body.appendChild(document.getElementById('div1').cloneNode(true));  
97 }  
98 }
```

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>JavaScript Validation API</title>
5         <script type="text/javascript">
6             function myFunction() {
7                 var inpObj = document.getElementById("id1");
8                 if (inpObj.checkValidity() == false) {
9                     document.getElementById("demo").innerHTML =
10                        inpObj.validationMessage;
11                 } else {
12                     document.getElementById("demo").innerHTML = "Input OK";
13                 }
14             }
15
16             function myFunction2() {
17                 var txt = "";
18                 if (document.getElementById("id1").validity.rangeOverflow)
19                     txt = "Value too large";
20                 else if (document.getElementById("id1").validity.rangeUnderflow)
21                     txt = "Value too small";
22                 document.getElementById("demo").innerHTML = txt;
23             }
24
25             function checkPasswords() {
26                 var pass = document.getElementById('pass');
27                 var confirmPass = document.getElementById('confirmPass');
28                 if (pass.value != confirmPass.value)
29                     confirmPass.setCustomValidity('הסיסמה לא תואמת');
30                 else
31                     confirmPass.setCustomValidity('');
32             }
33
34             function checkEmail(input) {
35                 //console.log(input.validity.customError);
36                 if (input == undefined)
37                     input = document.getElementById('email');
38                 if (input.validity.typeMismatch)
39                     input.setCustomValidity(input.value + "' is not a valid email. Enter something nice!!");
40                 else
41                     input.setCustomValidity("");
42                 //console.log(input.validity.customError);
43             }
44
45             function funcPattern() {
46                 if (document.getElementById('pat').validity.patternMismatch)
47                     alert("a-z or A-Z");
48             }
49     </script>
50 </head>
51 <body>
52     <p>Enter a number and click OK:</p>
53     <form>
54         <input id="id1" type="number" min="100" max="300">
```

appendchild
the end of file n/c

```
54      <input type="button" value="OK" onclick="myFunction()" />
55      <br /><input type="button" value="OK2" onclick="myFunction2()" />
56      <p>If the number is less than 100 or greater than 300, an error message will be displayed.</p>
57      <p id="demo"></p>
58      <div>
59          <label>password:</label><input type="password" id="pass" onchange="checkPasswords()" /><br />
60          <label>confirm password:</label><input type="password" id="confirmPass" onchange="checkPasswords()" />
61      </div>
62      <div>
63          <label>email:</label><input type="email" id="email" onchange="checkEmail(this)" /><br />תג' אונטראקיין
64          <label>email:</label><input type="email" id="email1" onchange="checkEmail(this)"/>
65      </div>
66      <div id="div5">
67          <label>pattern:</label><input type="text" name="pat" id="pat" pattern="[A-Za-z]+" title="a-z A-Z" onchange="funcPattern()" />
68          <br /><label>required:</label><input id="txtreq" type="text" required />
69          <br /><label>disabled:</label><input id="txt2" type="text" disabled />
70      </div>
71      <input type="submit" />
72  </form>
73 </body>
74 </html>
75
```

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <title>data attribute</title>
6
7   <script>
8     function start(){
9
10
11       document.getElementById('btn0').onclick = function () {
12         console.log(document.getElementById('myDiv0').getAttribute
13           ('data-my_data_description'));
14
15       document.getElementById('btn1').onclick = function () {
16         document.getElementById('myDiv0').setAttribute('data-
17           my_data_description', 'Some other string');
18
19       document.getElementById('btn1b').onclick = function () {
20         document.getElementById('myDiv0').removeAttribute('data-
21           my_data_description');
22
23       document.getElementById('btn2').onclick = function () {
24         var x=document.getElementsByTagName('div');
25         for(var i=0;i<x.length;i++)
26           x[i].setAttribute("data-greeting", "Hello World"+i);
27         console.log(document.getElementById('myDiv0').getAttribute
28           ('data-my_data_description'));
29         for (var i = 0; i < x.length; i++)
30           console.log(x[i].getAttribute('data-greeting'));
31
32       document.getElementById('btn3').onclick = function () {
33         console.log(document.getElementById
34           ('myDiv0').dataset.my_data_description);
35         document.getElementById('myDiv0').dataset.my_data_description =
36           "other string - dataset";
37         console.log(document.getElementById
38           ('myDiv0').dataset.my_data_description);
39
40         var x = document.querySelectorAll('[data-
41           my_data_description]');
42         var y = document.querySelectorAll("[data-greeting^='Hello
43           World']");
44
45         document.getElementById('myDiv0').dataset.my_data_description =
46           "";
47
48       }
49
50     }
51   </script>
52 </head>
```

```
46 <body onload="start()">
47
48 <h1>data-attribute</h1>
49 <div id="myDiv0" data-my_data_description="some string"></div>
50 <button id="btn0">Get data attached to div element - data-attribute</
      button><br />
51 <button id="btn1">Set data attached to div element</button><br />
52 <button id="btn1b">Removes data my_data_description attribute</button><br />
53
54 <h1>Attach Data to an Element</h1>
55 <button id="btn2">Attach data to div element</button><br />
56 <div></div>
57 <div></div>
58 <div></div>
59 <div></div>
60
61 <button id="btn3">Using JavaScript's dataset property</button><br />
62
63 <!--<div id="myDiv1" data-
      detail='{"name":"AAAAA","family":"Faaaa","age":10}'></div>-->
64 </body>
65 </html>
66
67
```

...ndow object\Window object2016\copy window object1.htm 1

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5   <title></title>
6   <script type="text/javascript">
7     var ww;
8
9     function f() {
10       //1
11       //var oNewWindow = window.open( [sURL] [, sName] [, sFeatures] [, bReplace]);
12
13 //window.open("copy window object2.htm");
14
15 //    window.open('copy window object2.htm', "", "width=400px,
16 //height=350px");
17
18 //2
19 //  document.location = 'copy window object2.htm';
20 window.location.href="copy window object2.htm";
21
22
23 //3
24 //  window.open('copy window object2.htm', '_self');
25
26
27 //4
28 // ww = window.open('copy window object2.htm');
29 //   ww.document.getElementById('t1').value = "abc";
30 //ww.document.body.onload = function () { ww.document.getElementById
31 //("t1").value = "abc"; ww.x = 20; ww.y = 50; ww.fl(66, 77); };
32
33 //  setTimeout(' ww.close();', 3000);
34
35 }
36
37
38
39
40 //2
41 function f2() {
42   alert("f2");
43   history.forward();//_self
44 }
45
46
47 function f3() {
48   // window.location = "view-source:" + window.location.href;
49 }
50
51 </script>
52 </head>
```

```
53 <body>
54   <input type="text" id='txt1' />
55   <input type="button" value="f()" onclick='f()' />
56   <input type="button" value="forward" onclick="f2()" />
57   <br />
58   <input type="button" id='bt3' value='הו יפה' onclick="f3()" />
59
60 </body>
61 </html>
62
```