גירסה 2.00 - 3/3/2002

DHTML

ניר אדר

מסמך זה מכיל הסברים אודות DHTML. ידע קודם נדרש להבנת המסמך הוא שליטה בשפת HTML. ידע ב-C/C++ או בשפת תכנות אחרת דרוש על מנת להבין את המושגים עליהם מדובר בנושא JavaScript. ההנחה היא שהקורא מכיר מושגים כגון פונקציות, משתנים וכדומה.

ה-DHTML בא לעזור לנו לעקוף מגבלות שונות שקיימות בHTML המקורי.
נציין מספר בעיות שאותן בא DHTML לפתור:
חוסר האפשרות לשנות את הדף לאחר שהוא כבר נטען. דפי HTML הם דפים סטטיים. הבדלים בין הדפדפנים השונים גורמים לאותו הדף להראות שונה בכל אחד מהדפדפנים.

שפת HTML איננה נותנת שליטה מלאה על המראה של הדף. HTML היא שפת הכוללת בעיקר הוראות מנחות לדפדפן איך להציג את המידע, ולא הוראות מדויקות. למשל, שורה אחת ארוכה שמופיעה על דפדפן אחד, יכולה להתפצל לשתי שורות בדפדפן שני.
אתרים גדולים הרוצים ליצור מספר רב של דפים, המעוצבים בסגנון דומה (צבע רקע, צבע כתב וכו') נאלצים לשכפל קוד רב. במידה ומתעורר הצורך לשנות את העיצוב של האתר, עבודה רבה נעשית על מנת להתאים את הדפים השונים לעיצוב החדש.

תוספות ה-DHTML הביאו איתם, בנוסף לפתרונות לבעיות לעיל, בעיות חדשות. יש הבדלים בין ה-DHTML שנתמך על ידי Internet Explorer לבין זה שקיים ב-Netscape או בMozilla. כמו כן, הדפדפנים הישנים אינם תומכים ב-DHTML. DHTML נתמכת במלואה על ידי גירסאות 4 ומעלה של הדפדפנים.

מסמך זה לא מתיימר להיות מדריך שלם לכל פקודות השפה. במסמך זה אתן סקירה על חלקים שונים מה-DHTML. ניסינו להדגיש את העיקר ולסנן חלק מהתכונות הפחות שימושיות למתכנת הממוצע.

חלק 1 - CSS

DHTML הוסיף פקודות חדשות והרחבות לHTML המקורי.

דוגמא הראשונה לכך היא התג <P>.

בHTML תג זה סימן התחלת פיסקה חדשה בלבד. DHML הוסיפה לתג זה פרמטרים נוספים.

דוגמא:

```
<HTML>
<BODY>
<P>
This is a plain paragraph.
<P STYLE="TEXT-INDENT: 30">
This paragraph is indented.
</P>
</BODY>
</HTML>
```

התוצאה תיראה כך :

This is a plain paragraph

This paragraph is indented.

השורה השנייה מוזחת מהתחלת הדף.

הדוגמא הבאה תדגים לנו אפשרות חדשה נוספת ב-DHTML:

```
<HTML>
<HEAD>
<STYLE>
P { COLOR: GREEN; TEXT-INDENT: 30; FONT-FAMILY: SANS-SERIF}
</STYLE>
</HEAD>
<BODY>
<P>
This is paragraph on a page with a global style.
</BODY>
</HTML>
```

התוצאה תיראה כך:

This is paragraph on a page with a global style.

הגדרנו ב- HEAD סגנון גלובלי לכל טקסט שיימצא בתוך התג <P>.
השורה מוזחת וירוקה וה FONT הוא SANS-SERIF. (בהנחה שה- FONT קיים במערכת).
נשים לב שהגדרנו סגנון עבור **כל** תג <P> שיופיע בדף.

אם נרצה להגדיר סגנון רק לפיסקה מסויימת, נוכל להשתמש בפרמטר STYLE. פרמטר זה נוסף לרוב התגים של HTML, ומבנהו הכללי משותף לכולם.
המבנה הכללי של STYLE הוא:

```
<TAG STYLE="style: value">  Blah  </TAG>
```

<u>דוגמא:</u>

```
<P STYLE="COLOR=GREEN"></P>
```

נשתמש בתג STYLE בתוך כותרת המסמך (<HEAD></HEAD>) על מנת להגדיר סגנון עבור כל הדף. הגדרת STYLE בתוך HEAD:

```
<HTML>
<HEAD>
      <STYLE>
      הגדרת הסגנון
      </STYLE>
</HEAD>
</HTML>

:דוגמא
<STYLE>
P { COLOR: BLUE; TEXT-INDENT: 30}
</STYLE>
```

ניתן לפצל את ההגדרה למספר שורות, על מנת שתהיה נוחה יותר לקריאה. הקוד הבא יפעל בצורה זהה:

```
<STYLE>
P { COLOR: BLUE;
       TEXT-INDENT: 30}
</STYLE>
```

לא כל דפדפן תומך ב-DHTML. על מנת שדפדפנים ישנים יתעלמו מהתגים ולא ידפיסו זבל, ניתן לשים "HTML COMMENTS" שיגרמו לדפדפנים הישנים יותר להתעלם מהתגים. הדפדפנים החדשים ינתחו את תגי הDHTML למרות ההערות.

דוגמא:

```
<STYLE>
<!--
P { COLOR: BLUE; TEXT-INDENT: 30}
-->
</STYLE>
```

DHTML תומכת גם בהגדרת סגנון אחד למספר תגים בו זמנית.

נביט בדוגמא הבאה כדי לראות כיצד:

```
<HTML>
<HEAD>
<STYLE>
<!--
H1,H2,H3,H4,H5,H6 { COLOR: BLUE}
-->
</STYLE>
</HEAD>
<BODY>
<H1>This is Blue</H1>
<H3>This is Blue too!!!</H3>
</HTML>
```

והתוצאה:

This is Blue

This is Blue too!!!

קבענו בהצהרה אחת שכל הכותרות במסמך יהיו בצבע כחול.

הכלת סגנון על מספר דפים בו זמנית

נניח שיש לנו אתר בו דפים רבים, ונרצה להחיל סגנון אחד על כולם.
במקום להגדיר את הסגנון בכל דף בנפרד, ניתן ליצור דף אחד, שיכיל את הגדרת הסגנון עבור כל הדפים. בכל דף נוסיף הוראה לפנות אל הקובץ ההוא על מנת להגדיר את הסגנון.
היתרונות של ריכוז כל הסגנונות בקובץ אחד הם רבים. בין הבולטים בהם הוא הימנעות משכפול קוד עבור כל דף, ואפשרות לשנות את סגנון האתר במהירות.

דוגמא:

```
<HTML>
<HEAD>
<LINK REL=”stylesheet” HREF=”globalstyle.css” TYPE=”text/css”>
</HEAD>
<BODY>
<P>Hello World!!</P>
</BODY>
</HTML>
```

קובץ ההגדרות: (globalstyle.css)

```
P { COLOR: BLUE;
       TEXT-INDENT: 30 }
```

השורה

```
<LINK REL=”stylesheet” HREF=”globalstyle.css” TYPE=”text/css”>
```

אומרת לדפדפן לקחת את הגדרות הסגנון שלו מהקובץ globalstyle.css.
ישנה דרך נוספת לעשות את אותו הדבר בדיוק:

```
<HTML>
<HEAD>
<STYLE TYPE="text/css">
   @import URL("globalstyle.css");
</STYLE>
</HEAD>
<BODY>
<P>Hello World!!</P>
</BODY>
</HTML>
```

התוצאה מבחינתנו תהיה זהה.
ב- Internet Explorer, השיטה השנייה היא היחידה שתעבוד.

דוגמא:

```
<STYLE>

BODY {FONT-FAMILY: sans-serif;
        COLOR: green;
        TEXT-ALIGN: justify;}
</STYLE>
```

בדוגמא זו הגדרנו את הסגנון של התג <BODY>.
כל מה שיהיה תחת BODY יקבל את הסגנון הזה, כולל תגים שתחת BODY, כלומר, נגיד בדף קיים טקסט אחרי התג <P>, הוא גם יקבל את הסגנון שהוגדר עבור ה- BODY, כלומר FONT -sans serif בצבע ירוק ומיושר.

בתוך התג <STYLE> יכולים לבוא פרמטרים רבים.
הנה רשימה של חלק מהם, ממוינים לנושאים:

צבע וצבע רקע

- color – קובע את צבע האובייקט עליו יחול הסגנון
- background-color – קובע את צבע הרקע של האובייקט
- background-image – קובע תמונת רקע לאוביקט

גופן

- font-size – קובע את גודל הגופן
- font-weight – קובע את עובי הגופן (דק, רגיל או מודגש)
- font-style – קובע את סגנון הגופן, כגון מודגש או מוטה
- font-family – מאפשר לך לקבוע משפחת גופנים מהם יבחר גופן למסמך, כגון serif
- color – קובע את צבע הגופן

מלל

- word-spacing – קובע את הרווח שיופיע בין המילים בדף
- letter-spacing – קובע את הרווח בין האותיות בדף
- text-align – קובע יישור של הטקסט לצד מסוים, כמו יישור לימין, לשמאל וכו'
- text-indent – קובע הזחה של הטקסט מתחילת השורה

קופסא

הכוונה בקופסא היא לאובייקטים הנמצאים מטיבם בתוך קופסא כגון תמונות, תאי טבלה וכדומה.

- width – קובע את רוחב הקופסא
- height – קובע את גובה הקופסא
- border-width – קובע את עובי גבולות הקופסא

יחידות שבהן ניתן להשתמש ב-STYLE:

סימון	יחידה
Pt	Points
Pc	Picas
Ex	X-height
Em	M-width
Mm	Millimeters
Cm	Centimeters
In	Inches
Px	Pixels

דוגמא:

```
<HTML>
<HEAD>
<STYLE>
     P { LETTER-SPACING: 2pc }
</STYLE>
</HEAD>
<BODY>
<P>
SPACE
</BODY>
</HTML>
```

והתוצאה:

S P A C E

כשרצינו להגדיר קודם צבע, כתבנו כך:

```
P { COLOR: BLUE }
```

ישנן דרכים נוספות להגדיר את הצבע בו אנו מעויינים:

```
P { COLOR: RGB(0, 0, 9) }
```

ודרך נוספת:

```
P { COLOR: RGB(100%, 0%, 100%) }
```

בדרכים אלו נוכל להגיע למגוון רחב יותר של צבעים.טבלה המציגה שוב את הפרמטרים של STYLE הקשורים לגופנים, ומראה מהם הערכים האפשריים לכל אחד מהפרמטרים:

פרמטר	ערכים מקובלים
FONT-SIZE	מספר, או אחד מהבאים: xx-small, x-small, small, medium, large, x-large, xx-large
FONT-FAMILY	serif, sans-serif, cursive, fantasy, monospace
FONT-STYLE	normal, italic, oblique, bold
COLOR	צבע

מחלקות

ייתכן שבאותו הדף, נרצה להגדיר יותר מסגנון אחד לפסקאות, כותרות וכו'. ייתכן שאת חלק מהכותרות נרצה לעצב בצורה אחת, וחלק אחר בצורה שונה. עדיין נרצה את האפשרות לרכז את העיצובים השונים במקום אחד, על מנת לשנותם בקלות.

כדי לבצע זאת באות לעזרתנו המחלקות.

בתוך תג ה-STYLE ניתן להגדיר מחלקות.

דוגמא:

```
<HTML>
<HEAD>
<STYLE>
H1.classname {COLOR: red}
</STYLE>
</HEAD>
<BODY>

<H1 CLASS=classname>This head would be red</H1>

</BODY>
</HTML>
```

הכותרת תוצג בצבע אדום.

בעזרת הפרמטר CLASS בחרנו אילו מבין העיצובים האפשריים שהגדרנו עבור H1 יבחר על מנת לעצב את הכותרת.

ניתן ליצור גם מחלקה שלא תשויך לתג מסוים, ויהיה ניתן להשתמש בה בכל אחד מהתגים.

דוגמא:

```
<HTML>
<HEAD>
<STYLE>
.specialtext {COLOR: blue}
</STYLE>
</HEAD>
<BODY>

<H1 CLASS=specialtext>This head would be blue</H1>
<P CLASS=specialtext>This text will be blue too</P>
</BODY>
</HTML>
```

שתי השורות יהיו בצבע כחול.

מספר המחלקות שניתן להגדיר עבור דף בודד אינו מוגבל.

דוגמא:

```
<HTML>
<HEAD>
<STYLE>
P.REDTEXT { COLOR: red }
P.GREENTEXT { COLOR: green; FONT-SIZE: 25 }
.BLUE { COLOR: blue }
</STYLE>
</HEAD>
<BODY>
<P CLASS=REDTEXT>This text will be red</P>
<P CLASS=GREENTEXT>This text will be green and larger</P>
<P CLASS=BLUE>This text will be blue</P>

</BODY>
</HTML>
```

מחלקות מדומות

מוגדרות ב- CSS גם מספר "מחלקות מדומות", מחלקות שהוגדרו מראש, שניתן להשתמש בהם בדפים שאנו יוצרים. קישורים הנמצאים בדפי HTML הם דוגמא למחלקה כזו. אנחנו יכולים לשלוט גם במחלקות אלו דרך תג ה- STYLE.

דוגמא:

```
A:link {COLOR: green}
A:active {COLOR: yellow}
A:visited {COLOR: gray }
```

בפקודות אלו אנו קובעים שכל הקישורים במסמך יהיו בצבע ירוק. כאשר לוחצים עליהם הם ישנו זמנית את צבעם לצהוב. אחרי שכבר ביקרנו בהם הם יהיו בצבע אפור.

שתי מחלקות נוספות הן first-letter ו-first-line, המתייחסות לאות הראשונה בקטע מסוים ולשורה הראשונה בקטע מסוים בהתאמה.

דוגמא:

```
<HTML>
<HEAD>
<STYLE>
P.TAGLINE:FIRST-LINE { COLOR: red }
P.DROPCAP {FONT-SIZE: 200% }
</STYLE>
</HEAD>
<BODY>
<P CLASS=TAGLINE>The first line in this paragraph will
be red</P>
<P CLASS= DROPCAP>The first letter in this paragraph will be
larger</P>
</BODY>
</HTML>
```

חלק 2 – JavaScript

על מנת ליצור דפים דינמיים, צריכה להיות דרך בה הנתונים שמקיש המשתמש יוכלו לעבור עיבוד, כך שיוצג פלט בהתאם לנתונים שהתקבלו.
ישנם מספר פתרונות אפשריים על מנת לייצר דפים דינמיים.
פתרון אחד הוא ניתוח בצד השרת. המשתמש שולח לשרת בקשה, והשרת מייצר דף HTML מותאם אישית עבור כל משתמש. שפות המשתמשות בגישה זו הן CGI, ASP ועוד. DHTML מציע פתרון נוסף שהוא סקריפטים הפועלים בצד הלקוח. שתי השפות העיקריות המשמשות ליצירת סקריפטים הינן JavaScript ו- VBScript.

שפת סקריפטים היא מעין סוג מוגבל של שפת תכנות. היא מכילה אלמנטים משפות תכנות, כגון תגובה למשתמש, לולאות, פונקציות, קלט ופלט, אולם יכולתה מוגבלת. השפה הוגבלה על מנת להתאים ל- Web. אפשרויות כגון יצירת קבצים, מצביעים, הוצאו מהשפה. ישנה בעייתיות בשימוש בסקריפטים. למרות שמתכנני הדפדפנים ניסו להגדיר שפה שבה המתכנת אינו יכול לכתוב דפים שיזיקו לאנשים הצופים בהם, אנשים שונים מצאו פירצות בהגדרות השפה, שאיפשרו להם להזיק למחשבי אנשים. אי לכך, אצל אנשים רבים, אפשרות הרצת הסקריפטים מבוטלת, למטרות אבטחה.
למרות שמתנהל מאבק בלתי פוסק לגרום ליצירת שפה ללא פרצות, עדיין המאבק לא הסתיים, וכאשר אתה בוחר האם להוסיף סקריפטים לדף שאתה יוצר, עליך לשקול מי הולך לצפות בו והאם הדפדפן שלו יתמוך בסקריפטים.

נתחיל מדוגמה פשוטה של JavaScript.

```
<HTML>
<HEAD>
<TITLE> Hello World </TITLE>
</HEAD>
<BODY>

<SCRIPT language="JavaScript">
   document.write("Hello, World");
</SCRIPT>
</BODY>
</HTML>
```

הפלט של סקריפט זה יהיה:

Hello, World

השורה <SCRIPT language="JavaScript"> אומרת לדפדפן שהקטע הבא הוא Script והשפה שלו היא JavaScript.

document.write("Hello, World"); שורה זו אומרת לדפדפן לכתוב Hello, World בחלון המסמך של הדפדפן.

ניתן לשלב בשורה זו גם תגי HTML.

דוגמא:

```
<SCRIPT language="JavaScript">
document.write("<H1> Hello, World </H1>");
</SCRIPT>
```

השורה </SCRIPT> מודיעה על סיום קטע הסקריפט.

מרכיבי JavaScript

לכל שפה יש דברים המרכיבים אותה.

דברים המרכיבים JavaScript הם הצהרות, בלוקים, הערות, נתונים, ביטויים ועוד.

הצהרה היא צירוף של פריט אחד או יותר בשורה אחת. דוגמא להצהרה:

```
document.write("Hello, World");
```

בלוק הוא קטע התחום בין סוגריים מסולסלים.

{

זהו בלוק

}

הערות הן קטעים מהן הדפדפן מתעלם.

ישנם שני סוגי הערות:

1. הערות של שורה אחת:

הערות אלו מסומנות על ידי // לדוגמא:

```
// This is comment
```

2. הערות של מספר שורות, מתחילות בסימן /* ומסתיימות ב/* לדוגמא:

```
/*
This is multi line comment.
Line 2 of the comment
*/
```

ב- JavaScript, כמו בשפות אחרות ישנם מספר סוגים של נתונים:

סוג נתון	ערכים לדוגמא
String	"Hello", "a String"
Number	12, 373.33
Boolean	true, false
Object	document, window
Function	sum(), addName()
Null	Null

ביטויים הם סידרה של משתנים ואופרטורים ביניהם.

דוגמא לביטוי:

```
(23-5)*3/7
```

ביטויים לוגיים וביטויי השוואה

אופרטור	שם	שימוש
&&	ו	(ביטוי1 && ביטוי2) מחזיר אמת אם שני הביטויים הם אמת
||	או	(ביטוי1 || ביטוי2) מחזיר אמת אם אחד משני הביטויים הוא אמת
!	לא	!(ביטוי1) מחזיר אמת אם הביטוי שקר
==	שווה	(ביטוי1 == ביטוי2) מחזיר אמת אם שני הביטויים הם שווים
!=	לא שווה	(ביטוי1 != ביטוי2) מחזיר אמת אם הביטויים אינם שווים
>	גדול	(ביטוי1 > ביטוי2) מחזיר אמת אם הביטוי הראשון גדול מהשני
>=	גדול או שווה	(ביטוי1 >= ביטוי2) מחזיר אמת אם הביטוי הראשון גדול או שווה לשני
<	קטן	(ביטוי1 < ביטוי2) מחזיר אמת אם הביטוי הראשון קטן מהשני
<=	קטן או שווה	(ביטוי1 <= ביטוי2)

		מחזיר אמת אם הביטוי הראשון קטן או שווה לשני

משתנים מאפשרים לך לשמור נתונים לשימוש מאוחר יותר בסקריפט.

הגדרת משתנה נעשית ב- JavaScript בצורה פשוטה, באמצעות שימוש במילת המפתח var.

דוגמא:

```
var position = 10;
```

בהצהרה זו יצרנו משתנה בשם position וקבענו לתוכו את הערך 10.

כאשר בוחרים שם למשתנה יש להתחשב בכללים הבאים:

שם המשתנה אינו יכול להיות מילה שמורה של JavaScript (לדוגמא אסור ליצור משתנה בשם var).

שם המשתנה חייב להתחיל באות אלפביתית או בתו _ .

הסימנים אחרי האות הראשונה יכולים להיות אותיות, מספרים או _.

המילים השמורות של JavaScript

abstract	boolean	break	Byte	case
catch	char	class	Const	continue
default	delete	do	Double	else
extends	false	final	Finally	float
for	function	goto	If	implements
import	in	instanceof	Int	interface
long	native	new	Null	package
private	protected	public	Return	short
static	super	switch	synchronized	this
throw	throws	transient	True	try
typeof	var	void	While	with

כאשר אתה נותן שם למשתנה, ישנה חשיבות לגבי אותיות גדולות וקטנות.

אלו למשל שני משתנים שונים לחלוטין:

```
var Variable;
var variaBle;
```

כדי לשנות את הערך של משתנה, כותבים:

ערך חדש = משתנה;

כדי למשל להגדיל את ערך המשתנה ב-10, ניתן לכתוב:

```
var=var+10;
```

פונקציות

ב- JavaScript ישנן פונקציות.

פונקציה היא קטע תוכנית המבצע פעולות על קבוצת נתונים מסוימת ומחזיר תוצאה. ישנן פונקציות הבנויות בתוך JavaScript וישנן פונקציות שהמשתמש יוצר.

דוגמא

דוגמא לפונקציה הקיימת ב- JavaScript , המחשבת את הערך של מחרוזת שהיא תרגיל חשבוני:

```
<HTML>
<HEAD>
      <TITLE>Eval Example</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
      var convertMe = "2+2";
      convertMe=eval(convertMe);
      document.write(covertMe);
</SCRIPT>
</BODY>
</HTML>
```

הערך של 2+2 יחושב ובדף תופיע הספרה 4.

חוץ מהפונקציות הבנויות בתוך JavaScript, המתכנת יכול ליצור פונקציות חדשות משלו, בצורה:

```
function myFunction(myText)
{

}
```

כדי שהפונצקיה תבצע משהו, נכתוב את הפעולות שנרצה שהפונקצית תבצע בין הסוגריים המסולסלים

דוגמא:

```
function myFunction(myText)
{
myText=eval(myText);
document.write(myText);
}
```

נדגים דף שלם המשתמש בפונקציה זו:

```
<HTML>
<HEAD>
      <TITLE>Function Example</TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
function myFunction(myText)
{
myText=eval(myText);
document.write(myText);
}
      var convertMe = "2+2";
      myFunction(convertMe);
</SCRIPT>
</BODY>
</HTML>
```

התוצאה תהייה זהה לתוצאה של הדוגמא הקודמת.

פונקציה יכולה להחזיר ערך. כדי לגרום לפונקציה להחזיר ערך משתמים במילה השמורה return.

דוגמא:

```
function square(inNum)
{
      return (inNum * inNum);
}
```

בהמשך נוכל להשתמש בערך של הפונקציה בצורה הבאה:

```
var squared;
squared = square(4);
```

הערך שיושם ב-squared יהיה 16.

הרעיון של הפיכת דפים לדינמיים היא בדרכים רבות קבלת החלטות לגבי הצגת הדפים, בהתאם לקלטים שונים. על מנת להשיג זאת אנו משתמשים בפקודות שונות.

1. if

זוהי הפקודה הבסיסית ביותר.

הצהרת if נבנית על ידי המילה if ואחריה ביטוי לוגי בין מרכאות, לאחר מכן באה הצהרה או בלוק, המתקיים אם ערך הביטוי הלוגי הוא אמת. דוגמא:

```
var x=5;
var y=15;
if ( x < y )
   document.write("x less than y");
if ( x == y )
   document.write("x equals y");
if ( x > y )
   document.write("x greater than y");
```

במקרה כזה, ייכתב בדף x less than y.

כל אחד מהתנאים שבביטויי הif נבדק. מכיוון שרק הביטוי הראשון הוא אמת, רק ההצהרה document.write("x less than y"); בוצעה.

בעזרת הביטוי הלוגי OR ניתן לגרום שהצהרה תתקיים אם אחד משני ביטויים יהיה אמת, בצורה הבאה:

```
if (( x == y ) ||  ( x < y )) {
     document.write("x less than y");
     document.write("or x equals y");
}
```

משמעות הביטוי ((x < y) || (x == y)) היא " x שווה ל-y או x קטן מ-y".

2. for

לולאות for אלו הלולאות הבסיסיות ביותר.

בלולאות for ישנו משתנה מונה שבעזרתו נקבע כמה פעמים הלולאה תתרחש, ישנו תנאי שאומר עד מתי הלולאה תתבצע וישנו ביטוי שמורץ בכל סבב של הלולאה.

נדגים את זה:

```
for (var count = 1; count <= 10; count++) {
  document.write(count);
  document.write("<BR>");
}
```

מה שקורה בשורות אלו:

מוגדר משתנה count וערכו נקבע ל- 1.

מכיוון שcount קטן מ- 10 ברגע זה, מבוצע הבלוק הפנימי

```
document.write(count);
document.write(“<BR>”);
```

מוצג בדף המספר 1.

אחר כך מתבצע count++. כרגע ערכו של count הוא 2.

שוב חוזרים לתנאי count <= 10. count עדיין עומד בתנאי וכך הלאה. הלולאה נפסקת כשcount גדול מ10. בדף שלנו יוצגו כל המספרים מ- 1 עד 10.

3. while

לולאות while דומות ללולאות for, אבל יש בהן רק את התנאי שצריך להתקיים ולא את כל שאר החלקים. בדרך כלל מוגדר המשתנה שאחראי לסיום הלולאה לפני הלולאה עצמה.

דוגמא:

```
var count = 1;
while (count <= 10)
{
   document.write(count);
   document.write(“<BR>”);
   count++;
}
```

לולאה זאת עושה בדיוק את אותה הפעולה שעושה הלולאה הקודמת. הבחירה בין הלולאות בזמן בניית הדף נעשית לפי צרכי המתכנת.

אובייקטים ב- JavaScript

תכנות מונחה עצמים הוא אחד הנושאים הפופולריים ביותר בתחום המחשבים.

כל השפות הפופולריות ביותר בימים אלו מכילות תכנות מונחה עצמים במידה זו או אחרת.

JavaScript אינה שונה משפות אחרות בנושא זה.

בניגוד לשפות כגון C++ ,javaı JavaScript אינה תומכת בתורשה ורב צורתיות. JavaScript תומכת רק בתכונת הריכוזיות של התכנות מונחה העצמים. תכונת הריכוזיות היא האפשרות לתת לקבוצה של משתנים ושל פונקציות שם משותף, להפוך אותם למעין חבילה אחת בשם אובייקט. בחבילה זו המשתנים נקראים "תכונות" האובייקט, והפונקציות נקראות "שגרות" האובייקט או "שיטות" האובייקט. ניקח דוגמא כדי להסביר את מושג הריכוזיות. נגיד למשל שאתה רוצה לייצג מכונית בעזרת אובייקט.

למכונית ישנן מספר פונקציות, למשל

- Push_Gas()
- Run_PushBrake()
- SteerRight()

ועוד.

נניח שקראת לאובייקט המחשב MyCar, אתה יכול לגשת לשיטות שלו בצורה הבאה:

```
MyCar. Push_Gas();
```

המידע של המחשב שלנו יכול להישמר ב"תכונות" שלו.

דוגמא לשימוש בתכונה:

```
MyCar.speed = 25;
```

ב- JavaScript ישנם אובייקטים בנויים בתוך השפה. דוגמא לאובייקט היא Math.

אחד מהביטויים המתמטיים הנפוצים שאובייקט Math מכיל הוא PI.

דוגמא לפונקציה שמשתמשת ב- PI, על מנת לחשב שטח מעגל:

```
function area(radius) {
     return (radius * radius * Math.PI);
}
```

דוגמא לשיטה הקיימת בתוך אובייקט ה- Math:

```
var myNumber = Math.sqrt(16);
document.write(myNumber);
```

דוגמא זו תציג על הדף את השורש הריבועי של המספר 16. השתמשתי בשיטה sqrt שבאובייקט Math. שיטות נוספות הקיימות באובייקט Math:

- abs – השיטה מקבלת מספר ומחזירה את הערך המוחלט של המספר.
- cos, sin, tan – הפונקציות המתמטיות המוכרות.
- acos, asin, atan – הפונקציות המתמטיות arccos, arcsin, arctan.
- max – מקבל שני מספרים ומחזיר את הגדול ביניהם.
- min – מקבל שני מספרים ומחזיר את הקטן ביניהם.
- pow – מקבל מספר בסיס ומספר מעריך ומחזיר את החזקה של הבסיס עם המעריך.
- random – השיטה אינה מקבל כל ערך. היא מחזירה מספר אקראי בין 0 ל1.
- round – מעגל את המספר הנתון לערך השלם הקרוב ביותר.

אובייקט נוסף המובנה בתוך JavaScript הוא String.

כל מחרוזת ב- JavaScript היא למעשה אובייקט. למחרוזת יש תכונות ושיטות היכולות להיקרא כדי לנתח ולשנות את המחרוזת בדרך זו או אחרת.

דוגמא:

```
<HTML>

<HEAD>
<TITLE></TITLE>
</HEAD>
<BODY>
<SCRIPT language="JavaScript">
  var message = "Hello World!!!<BR>"
  document.write(message);
  document.write(message.toLowerCase());
  document.write(message.toUpperCase());
  document.write("Welcome to my page".toUpperCase());
</SCRIPT>
</BODY>
</HTML>
```

()toLowerCase היא שיטה ההופכת את כל המחרוזת לאותיות קטנות.

()toUpperCase היא שיטה ההופכת את כל המחרוזת לאותיות גדולות.

בדוגמא לעיל, הפלט יהיה:

Hello World!!!

hello world!!!

HELLO WORLD!!!

WELCOME TO MY PAGE

דוגמא לפונקציה נוספת:

```
var warning = “Death to all who enter here”
document.write(warning.indexOf(“enter”, 0))
```

על הדף ייכתב 17, המיקום של המילה enter בתוך המחרוזת warning.

הפונקציה substr מאפשרת לנו לגזור חלקי מחרוזות ולהציגן או לבצע עליהן פעולות.

דוגמא:

```
var story = "Long time ago..."
document.write(story.substr(10, 3))
```

בדף תופיע המילה ago. הפרמטר הראשון של הפונקציה מציין את התו הראשון ממנו מתחילים לגזור, והפרמטר השני מציין כמה תווים לגזור מהמחרוזת.

מספר פונקציות של מחרוזות מוסיפות תגים של HTML למסמך.
הנה רשימה של פונקציות אלו, והתגים אותן הן מציבות:

תג	פונקציה
<A NAME="…">	anchor(name)
<A HREF="…">	link(url)
<BIG>	big()
<SMALL>	small()
<BLINK>	blink()
<B>	bold()
<I>	italic()
<FONT COLOR="…">	fontcolor()
<FONT SIZE="…">	fontsize()

דוגמאות:

```
var story = "Long time ago...<BR>"
document.write(story.link("http://start.at/uw2000"))
document.write(story.fontcolor("#ff44ff"))
```

ישנם אובייקטים נוספים מובנים בתוך השפה, אולם כעת לא נתרכז בהם.

כעת נרצה כיצד ניתן ליצור אובייקטים חדשים.
הדרך ליצירת אובייקטים היא עקיפה. לא ניתן ליצור אובייקטים באופן ישיר.
כדי ליצור אובייקטים עליך לכתוב פונקציה שמכילה את כל התכונות והשיטות של האובייקט. לאחר מכן מגדירים את התכונות של האובייקט בעזרת המילה השמורה this. לדוגמא, נניח שאתה רוצה ליצור אובייקט שייצג ספר, אתה יכול להגדיר את האובייקט כך:

```
function book(name, author, releasedate)
{
      this.name = name;
      this.author = author;
      this.releasedate = releasedate;
}
```

בשלב זה לא יצרנו את האובייקט אלא יצרנו את הפונקציה שיוצרת את האובייקט.

עכשיו ניצור למשל אובייקט מסוג ספר:

```
var MyBook
MyBook = new book(“Sumsum Street”, “Moshe Ofnik”, 1982);
```

כעת נוכל להתייחס לתכונות האובייקט:

```
document.write(MyBook.name);
```

מערכים

נניח שנרצה לאכסן רשימה של 25 מספרים.

דרך אחת לבצע זאת היא ליצור 25 משתנים שיכילו את המספרים.

דרך שניה היא ליצור מערך שיכיל אותם. מערך הוא בעצם קבוצת משתנים, להם שם אחד, והנבדלים זה מזה על ידי אינדקס.

מערך נוצר בדרך דומה לאובייקט, וביצירתו מגדירים את מספר המשתנים שתזדקק להם.

דוגמא

```
MyNumbers = new Array(25);
```

ניתן לגשת לאיברים של המערך על ידי כתיבת שם המערך ומספר בסוגריים מרובעים. למשל, על מנת לגשת לאיבר הראשון במערך, ולהציב בו ערך, עליך לכתוב:

```
MyNumbers[0] = 1234;
```

על מנת לגשת לאיבר השני כתוב:

```
MyNumbers[1] = 2345;
```

וכך הלאה.