

INTERNET BASED PROGRAMMING

Active Server Pages (ASP)

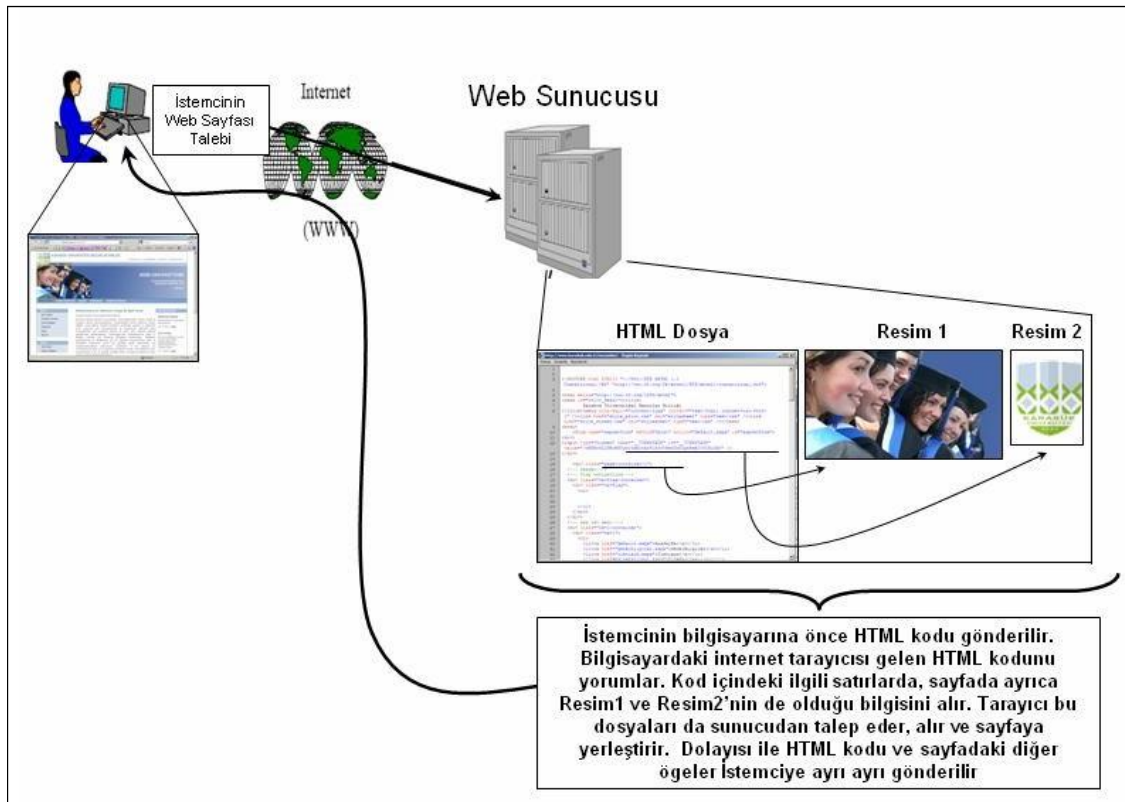
Programming with Vbscript

Prof. Dr. İsmail Rakıp Karaş

ASP Technology

Introduction

As you can remember from the previous lessons, files with HTML extension (HTML codes) were sent to the client by the server and these codes were interpreted / executed by the browser on the client's computer. (server, client, browser, HTML files, interpret, interpretation, running)



Transfer of HTML codes from the server to the client and interpretation / execution by the browser on the client's computer

In this approach, the server is not concerned with HTML codes, the server is only obliged to send these codes to the client. However, this approach is often inadequate. Interactive and dynamic pages running on the server side are needed instead of static pages running on the client side. On top of that, for security, inadequacy of browsers and reducing client load, some operations are required on the server side (security, inability of browsers). Technologies and languages such as PHP, ASP, ASP.Net are some server-side solutions that perform such operations. This course will focus on ASP technology.

Fat and Thin Client Approach

As in the example of the HTM files above, if the transactions are performed on the client's computer, this approach is called the Fat Client approach. The reason for being called fat is because the transaction load is mostly on the client side. In contrast to this approach, when the transaction load is on the server side, this approach is called the Thin Client approach (Server side approach. In this case, the processing load on the client is low, that is, weak. The server is obliged not only to send the data but also to interpret, run and evaluate the relevant codes before sending.

A cartoon from the early stages of the Thin Client approach:

1.



IT Staff

2.



(Welcome to Weight Hunters Association Sessions.)
- I can't stay away from chocolate
- Damn...

Two Examples:

Let us examine the examples below in order to better understand and embody the issue. Note that there are different lines (red lines) in both examples that start with tags that we don't recognize within HTML codes. These lines are script code. The lines in the example on the left are JavaScript, the ones on the right are VbScript code.

Merhaba.htm

```
<HTML>
<HEAD>
<TITLE>JavaScript ile Tarih</TITLE>
<meta http-equiv="content-type"
content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1254">
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>Merhaba Dünya</H1>
<H2>Bugün:</H2>
<H3>
<SCRIPT LANGUAGE=JAVASCRIPT>
<!--
tarih = new Date();
document.write(tarih);
//-->
</SCRIPT>
</H3></BODY>
</HTML>
```

Merhaba.asp

```
<HTML>
<HEAD>
<TITLE>ASP İLE İLK SAYFA</TITLE>
<META http-equiv="content-type"
content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type"
content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H1><CENTER>Merhaba Dünya!</H1>
<H2>Bugün:
<%
Response.Write(WeekdayName(WeekDay(Date))
& " " & Date & " " & Time)%>
</CENTER>
</H2>
</BODY>
</HTML>
```

The files containing these codes have been uploaded to a server so that you can see the differences between them better. Please enter the addresses below to see how these codes are displayed in your browser:

<http://www.ismailkaras.com/ASP/merhaba.htm>
<http://www.ismailkaras.com/ASP/merhaba.asp>

When you enter the pages, you will see that both codes give the date of the day similar to each other.

Now, examine the codes of the related pages using the "Show Source Code" feature of your browser. When you look at the code of the first address, you will see that the codes shown on the left above are written exactly here. When you look at the code for the second address, you will see that the date of the day is in the place where

Response.Write(WeekdayName(WeekDay(Date)) & " " & Date & " " & Time)%>

is different from the example above.

The reason for this is this; While JavaScript is a client-side (Fat client) language, VbScript is a server-side (Weak Client) language, on the contrary. Therefore, when you look at the Source code of the first page, the reason you can see the code is exactly that. The codes are working on your (client) side and you can see the source of these codes as they come to your computer. The VbScript code running on the right runs on the server and only the result of the transaction is sent to the client. Therefore, VbScript code does not appear in the Source code. Because this code has not come to your browser. It is the result of VbScript code that runs on the server, coming to your browser.

While these examples present the Weak-Fat User approach nicely, two other things should have drawn your attention to those described above.

- Script Languages are written with HTML codes (nested).
- The % sign stands out where the VbScript code starts.

VbScript Language

Contrary to the general misconception, ASP is not a language. ASP (Active Server Pages) is the name of a technology. As its name is, it is active / dynamic pages running on the server. The codes that work within these pages are VbScript codes. In other words, the language used by ASP technology is VbScript.

VbScript is a script language. In other words, it is a simplified form of Visual Basic language for web applications (Simplified VB). It is far from demonstrating the capabilities that Visual Basic language provides in desktop programming. However, it contains tools and methods that can be used effectively in web applications. The coding method (syntax) is very close to Visual Basic. It is very easy to pass from one to the other.

As noted above, VbScript codes can be used nested / embedded with HTML codes. It is understood with the <% tag that the VbScript code starts in the HTML code. Therefore, the start tag of the VbScript code is <% and the end tag is%>. The codes between these tags are perceived as data that should be interpreted by the server. The server sends the interpretation of the htm codes in the ASP file to the client, and sends the VbScript code and sends the results.

The extension of ASP pages is “.asp”. If the extension of a file saved on the server is asp, the server detects that it is a file that needs to be run and interpreted, and looks into it looking for VbScript codes. However, if the extension of the file is .htm, the server does not interpret this file in any way, it sends all of them to the client when requested. So even if a file has VbScript code inside it, the server will not look at the codes inside that file unless its extension is .asp. It will send directly to the client. Since the browser in the client cannot understand the VbScript codes, it will assume that the lines are not present and cannot be visualized.

As a result, VbScript codes in the same file are interpreted by the server, html codes and combined and visualized in the browser.

Example:

The extension of the file named “Merhaba.asp” above has been changed to htm and saved on the same server as “Merhaba_.htm”. Call the file to your browser by clicking the link below.

http://www.ismailkaras.com/ASP/merhaba_.htm

Unlike the previous one, you should have noticed that the date is not shown / visualized. Now, examine the codes from your browser using the "Show Source" feature. In contrast to the above situation, you will be able to see the Vbscript code that says

Response.Write (WeekdayName (WeekDay (Date)) & " " & Date & " " & Time)%>.

Since the file is htm, the server sends the file completely to the client without interpretation, and your browser assumed that these lines are ignored because they did not recognize VbScript codes. As a result, if we are writing VbScript code in a file, we must definitely convert the file's extension to .asp, which should be interpreted by the server.

When the client / user calls a file with the extension .asp (click / link to link), the server first looks at the file's extension. When the server recognizes that it is asp file, it starts to read the codes inside. It sends the htm codes exactly to the client without interpreting them. When it sees the <% tag, it detects the VbScript code has started and starts to interpret / run this code from that point on. It then sends the results obtained by executing the code to the client. When the tag is finished and returned to htm, it leaves the comment and sends the same code to the client.

Examples:

Click on the links of the examples below and compare the codes shown in your browser with the "Show Source" and the original codes below. See the differences. Try to understand the structure of ASP and VbScript language by examining and guess what happens on the server side. See the examples described above on the examples.

<http://www.ismailkaras.com/asp/gunler.asp>

```
<HTML>
<HEAD>
<TITLE>ASP GÜNLERİ SAYMA</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H2>
<CENTER>
<%
Dim Gun
Gun = Array("Trakya", "Ege", "Marmara", "Doğu Anadolu", "Güneydoğu Anadolu", "Akdeniz", "Karadeniz")
For sayac = 0 to 6
    Response.Write Gun(sayac)
    Response.Write "<BR>"
Next
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

<http://www.ismailkaras.com/asp/deneme.asp>

```
<% Option Explicit %>
<HTML>
<%
Dim Degisken(2), Toplam
Degisken(1) = "Mustafa"
Degisken(2) = "Durcan"
Toplam = degisken(1) + Degisken(2)

=Toplam %>
</HTML>
```

<http://www.ismailkaras.com/asp/hosgeldiniz01.asp>

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GÖRE SELAM</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H2>
<CENTER>
<%
If Hour(Now) <12 Then
    Response.Write "Günaydın! "
Elseif Hour(Now) >= 18 Then
    Response.Write "İyi akşamlar! "
Else
    Response.Write "Tünaydın! "
End If
Response.Write "<BR>"
Response.Write "Sitemize Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

<http://www.ismailkaras.com/asp/hosgeldiniz02.asp>

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GÖRE SELAM</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
```

```
<BODY>
<H2>
<CENTER>
<%
Select Case Hour(Now)
    Case 0,1,2,3,4,5,6,7,8,9,10,11
        Response.Write "Günaydın!"
    Case 12,13,14,15,16,17
        Response.Write "Tünaydın"
    Case Else
        Response.Write "İyi Akşamlar!"
End Select
Response.Write "<BR>"
Response.Write "Sitemize Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

Running ASP Files on Your Own Computer

Question: Do I have to upload the ASP files to a server? Can't I run it on my own computer? Can't I make my own computer act like a server or client?

Of course, you can use your own computer like an ASP server. For this, IIS (Internet Information Server) must be installed. You can access many resources on the Internet about IIS installation. However, we will use another program that is much simple to install in this lesson to try our ASP files:

Babyweb.exe

To use the program, follow these steps:

1. Download Babyweb.exe from <http://web.karabuk.edu.tr/ismail.karas/asp/> to anywhere.
2. Open a folder to store and try your ASP files (Ex: C: \ ASPexperiment).
3. Run Babyweb.exe. Click Settingse and enter the address of this folder on the line that says "Web Pages". (You can also show the path by clicking the three dots to the right.)
4. After saying "OK", take the window down and you will see that the program continues to work next to the clock.
5. The folder you specified is now your root folder / localhost on your server. All to be able to run. You should keep your asp files here. To run ASP files in this folder, you must:

a. For example, if your computer name is hakan1 (You can look at your computer name from "my computer> properties" (Full computer name). If your computer name contains Turkish characters and spaces, change its name.)

b. The file you will run is merhaba.asp. In this case, to run the file, you should write the address line of the browser as follows:

[http: //hakan1/merhaba.asp](http://hakan1/merhaba.asp)

Or you can reach your files in an easier and shorter way as follows:

[http: //localhost/merhaba.asp](http://localhost/merhaba.asp)

PROGRAMMING WITH VBSCRIPT

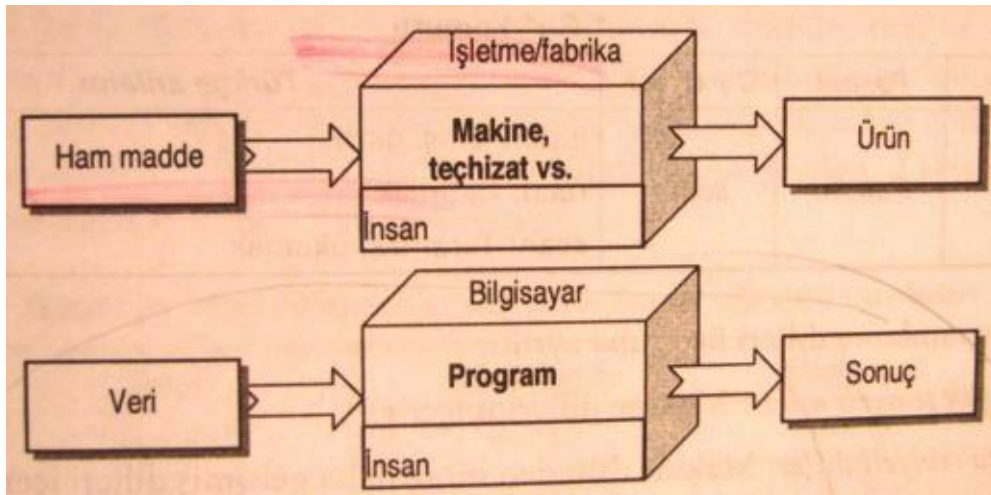
As mentioned above, VbScript is a script language based on the Visual Basic language. Therefore, as in all other languages, Programming principles are important in VbScript. In this respect, primarily the basics of Programming with VB will be emphasized.

FUNDAMENTALS OF PROGRAMMING

1. BASIC CONCEPTS

Computer Program and Programming

The **program**, which is a communication tool between human and computer; It is a series of commands transmitted to the computer in order to obtain desired output values by using input values. Computers need a program to process and solve problems. These programs, on the other hand, are reported to the computer by human beings or in other words, they are taught. This teaching process is called **programming**.



The program in the computer and the operation of the machine in the factory are similar

Algorithm

Whatever the user teaches the computer, it gets its response. With the programs it enters, it tells the computer what data to process, and what kind of results it will produce. In order for the user to do this, he must first know the solution of the problem. It is therefore important that the user plans how to do this before entering / teaching a program to the computer. This planning is done step by step and this process is called algorithm.

Therefore, the algorithm is a set of rules and steps to be followed in the execution of a transaction(s) on the computer. The user who prepares the algorithm writes the steps he / she wants to do from the computer in the language he / she understands (human language) step by step. Algorithm is a study plan that shows which stages of a work to be done. Algorithm is not a programming language.



The word "Algorithm" was a term made by westerners from the famous mathematician Mohammed bin Musa al-Harezmi, the word al-Harezmi, which Arabs gave him as yellow.

el-harezmi > algorizma >algoritma

A good algorithm should be designed to take action as soon as possible, to use the least amount of memory and to consume computer resources to a minimum.

When preparing an algorithm

- The work to be done is thoroughly examined. All possibilities are reviewed. (examined and considered all probabilities)
- The solution that will deliver the most accurate and sensitive result in the shortest time, in the least processing step, is determined.

To design an algorithm,

- Makes the program easy to write (make easier).
- Reduces the rate of bad encoding.
- Shortens the time it takes for program writing (timesaving).
- It facilitates program control since it clearly shows the process flow.
- It provides convenience in subsequent arrangements.

Example: Algorithm of Selling Tickets for Theater on the Internet:

1. The game, day and place information requested by the user is taken.
2. The database is queried available seats of the game played on the specified day.
3. If the number of available seats is zero, the game specified on that day is not played, or all the seats in the game are sold.
4. In both cases, the error message appears on the screen because the ticket could not be issued. Returned to the beginning to get the day and game information again.
5. The seat information is obtained from the user.
6. It is checked whether the seat is available or not.
7. If the place is not available, an error message is shown on the screen and returned to the 5th step to request the seat information.
8. If the seat is empty, the seat records of the game are updated in the database.
9. The ticket containing the desired day, game and seat information is issued from the printer.

Programming Languages

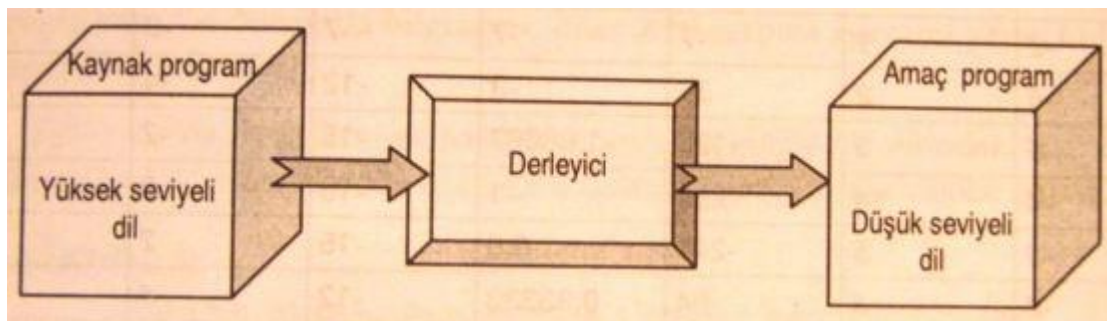
How do we get directions to someone who asks for a place using the words they understand (Turkish, English, etc.). When we ask the computer to do something, we need to use the languages that the computer understands. All of the special words, signs and symbols used by the user when entering / teaching a program are called programming language. Programming languages are generally divided into three groups.

1. Low-level languages: Contains machine languages. A very complex language of 0 and 1's
2. Intermediate languages (Mid-level): It is a slightly more advanced language than machine language.
3. High-level languages: Languages close to the human spoken language.

The programming languages commonly used today are high-level languages. And some of these are: Pascal, Cobol, Basic,, C ++, C #, Delphi, Visual C, Visual Basic etc.

(In this course, we will learn the language of Visual Basic and to write a program in this language because it is simple, understandable, and visual.)

In order for the program written in a high level language to work, it must be converted to the machine language (ie the file with the EXE extension). This conversion process is called compilation process.



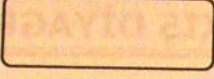
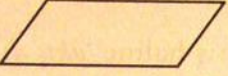
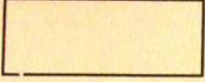
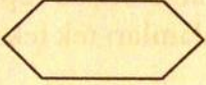

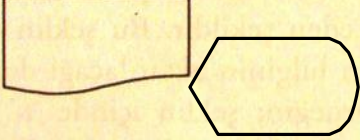

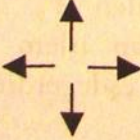
The compilation process

Program Coding and Program Source Code

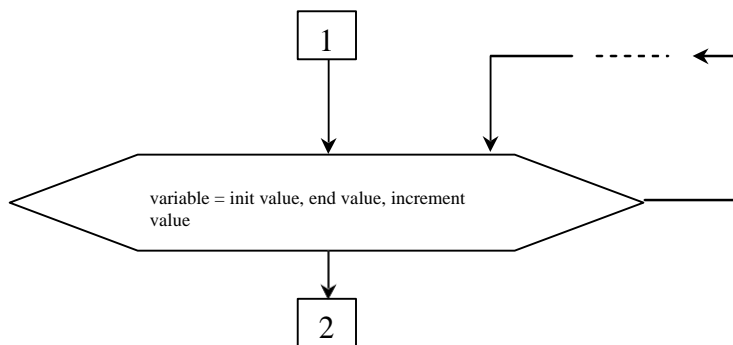
Writing the designed algorithm in a specific programming language using the command and terminology of that language is program coding, or coding, and this text is called program code or program source code.

2. FLOWCHARTS

The flow of algorithms drawn with special geometric shapes is called a flowchart diagram. The shapes used in flow diagrams and their meanings are as follows:

Şekil	Anlamı	Açıklama
	Başla / Dur	Specifies the points where the algorithm starts and ends in flow charts. (Start- Begin / Stop-Exit)
	Bilgi/veri girişi	It is expressed in this way when there will be an external information entry. Specifies to enter a value by the user. (Input / Data entering)
	İşlem	The operations to be performed during the program (eg mathematical operations) are expressed in this way. (Process)
	Döngü	Indicates that the program entered a certain cycle at this point. Details of the cycle are given below. (Loop)
	Karar/Karşılaştırma	This figure is used if a decision will be made by making a specific comparison and the direction of the algorithm will change accordingly. The number of arrows separated from this figure is more than one (Decision)
	Bilgi/veri yazma	This format should be used if an output or image notification will be made in peripherals such as display, printer and similar types. (Document / Output)
	Bağlantı	It is put at the junction points of the arrows in the flow diagrams. It is not compulsory. (No.)
	İşlem akış yönü	The arrows are used to join the shapes mentioned above and to show the directions of the flow. (Arrows)

Loops in Flowcharts



The way the loops in the flow diagrams work is as follows: The algorithm from path 1 enters the loop when you see the shape next to it. The variable specified here will start from the initial value, increment or decrement each time and will reach the end value. Once it reaches the end value, the loop will end and the algorithm will continue to run in ways 2. Below are examples of different cycles according to the start-end values and the increase or decrease amounts.

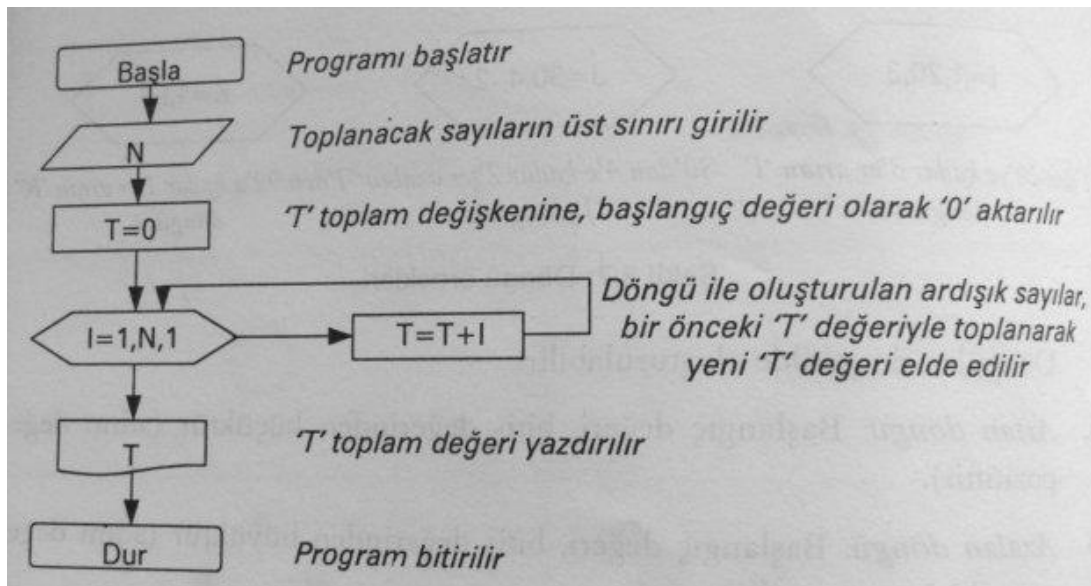
	<p>The variable i will start from 1 and will increase by 3 and continue until 20. That is, it will continue as $i = 1$ in the first cycle, $i = 4$ in the second, and then 7, 10, 13, 16, 19 in the following cycles. Then the cycle will end and the flow will continue down. During this cycle, the operations given in the dotted section for each value of i will be repeated.</p>
	<p>The variable j will start at 30 and will decrease by 4 (decreasing by twos) until 4. So, in the first loop, like $j = 30$, in the second, like $j = 28$. Then the cycle will end and the flow will continue down. During this cycle, the processes given in the dotted section for each value of j will be repeated.</p>
	<p>Variable k will start from 1 and will increase by 1 and continue to 99. So, $k = 1$ in the first cycle, $k = 2$ in the second. Then the cycle will end and the flow will continue down. During this cycle, the operations given in the dotted section for each value of k will be repeated. The point to be considered here is; If the amount of increase is not given, this value is always equal to +1.</p>

Various Algorithm Examples with Flowcharts

EXAMPLE 1: Preparing tea brewing process with algorithm and flow diagram

Algorithm:	Flowchart:
<ol style="list-style-type: none">1. The user is expected to give water.2. The water is heated.3. It is checked whether the water boils or not. If it does not boil, return to step 2.4. As the tea is already prepared, it is not expected from the user. Tea is put into the teapot.5. The user is informed how long the brewing process will take.6. The present time (when the tea starts to brew) with the brewing time received from the user is collected. The resulting value is assigned to the variable named End Time. This variable keeps track of when the brewing process will end.7. If the present time is less than the end time, the time allocated for the tea to brew has not expired yet. Step 7 is repeated until this time is up.8. A message appears on the screen informing the user that the tea is infused.	

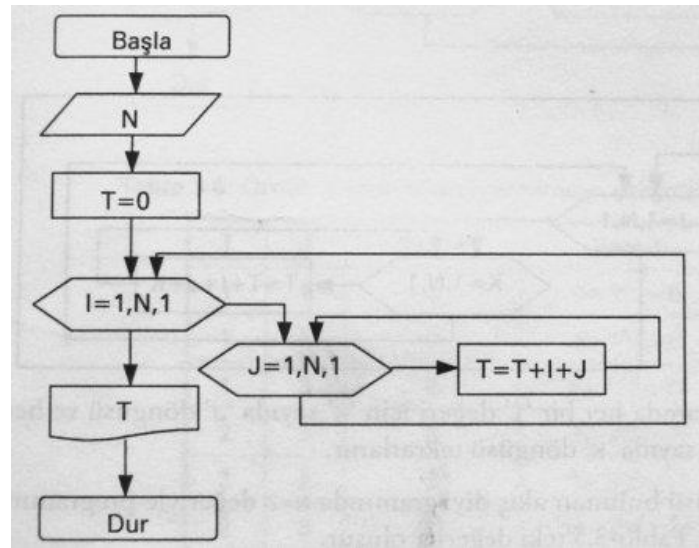
EXAMPLE 2 (Loops): Flow chart of the program that calculates the sum of numbers from 1 to N (sum of numbers one to n) and prints on the screen, **depending on the N value entered by the user**:



If the user entered 5 for the N value, the values that the variable i will take in each cycle of the cycle will be as follows:

I	Eski T	Yeni T
1	0	$0+1=1$
2	1	$1+2=3$
3	3	$3+3=6$
4	6	$6+4=10$
5	10	$10+5=15$

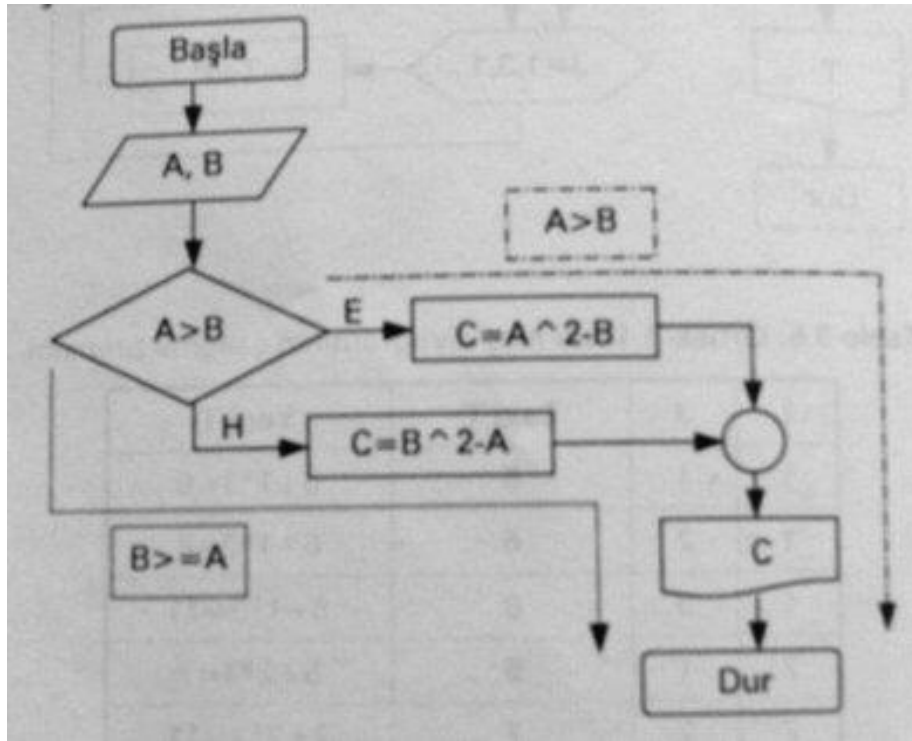
EXAMPLE 3 (Nested Loops): Flowchart of transactions with two nested loops, depending on the N value entered by the user:



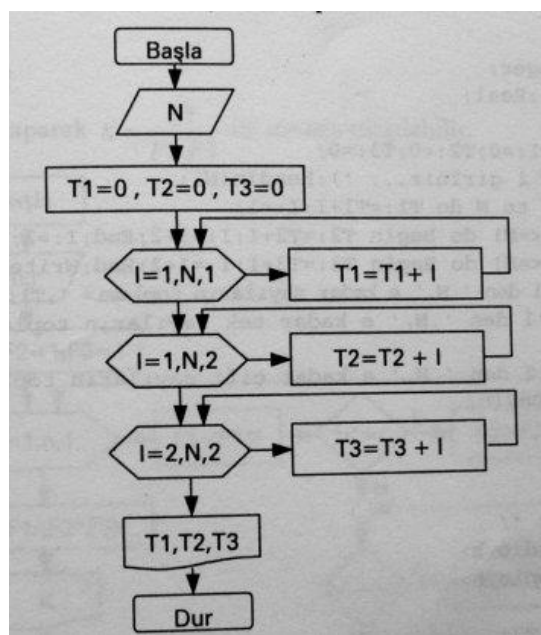
If the user entered 3 for the N value, the values that the i and j variables will take in each cycle of the cycle will be as follows:

I	J	Eski T	Yeni T
1	1	0	$0+1+1=2$
1	2	2	$2+1+2=5$
1	3	5	$5+1+3=9$
2	1	9	$9+2+1=12$
2	2	12	$12+2+2=16$
2	3	16	$16+2+3=21$
3	1	21	$21+3+1=25$
3	2	25	$25+3+2=30$
3	3	30	$30+3+3=36$

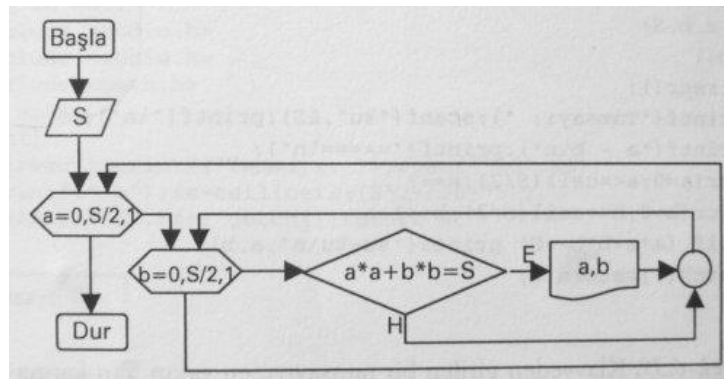
EXAMPLE 4 (COMPARISON-Decision): Flowchart of the program that compares the A and B values entered by the user and performs $C = A^2 - B$ or $C = B^2 - A$ operations accordingly and prints the result on the screen.



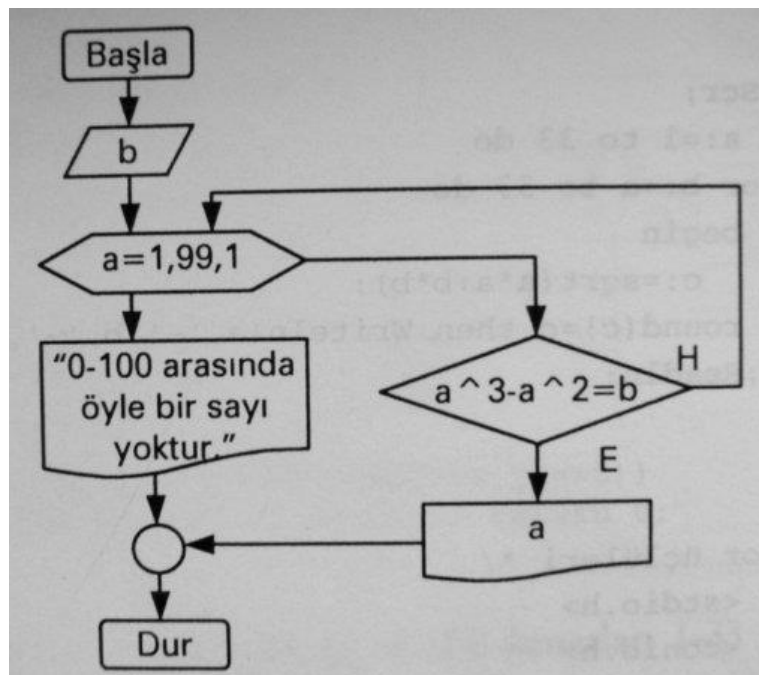
EXAMPLE 5 (CYCLE): Flow chart of the program that calculates and prints the sum of integers, odd numbers and even numbers from 1 to N, depending on the N value entered by the user (Attention to the initial and increment values of the loops)



EXAMPLE 5 (Comparison-Nested Loops and Decision): Flow chart of the program that gives the user whether the integer S entered can be written as the sum of squares of two numbers:



EXAMPLE 6 (CYCLE and COMPARISON): The flow chart of the program that finds the numbers between 0 and 100 satisfying the $a^3 - a^2 = b$ condition according to the integer b entered by the user:



3. BASIC OPERATIONS CARRIED OUT BY COMPUTER PROGRAMS

The processes performed with computer programs are generally divided into three:

1. Mathematical (arithmetic) operations
2. Comparison (decision) operations
3. Logical operations

3.1. Mathematical Operations

When performing mathematical operations on the computer, it must be written in a language appropriate to its own language. These statements should be as follows.

İşlem	Matematik	Bilgisayar
Toplama	$a + b$	$a + b$
Çıkarma	$a - b$	$a - b$
Çarpma	$a \cdot b$	$a * b$
Bölme	$a \div b$	a / b
Üs alma	a^b	$a \wedge b$

Mathematical operations and computer language equivalents

Order of priority mathematical order is very important. The computer performs the operations in the order specified in the table below. That is, in a mathematical expression, the computer first makes the brackets (pharanteses), then exponential expressions, then multiplication and division, and finally the addition and subtraction. If this order is not taken into consideration while writing the transactions, the results will be incorrect.

İşlem öncelik sıraları		
Sıra	İşlem	Bilgisayar dili
1	Parantezler	$((\dots\dots\dots))$
2	Üs alma	$a \wedge b$
3	Çarpma ve bölme	$a * b$ ve a / b
4	Toplama ve çıkarma	$a + b$ ve $a - b$

Mathematical operation priority orders

In the same priority, the order of operations is from left to right. Therefore, it is very important to pay attention to this while writing.

For example, in $A * B / C$ operation, the computer will first multiply A by B, then divide the output by C.

Examples:

Matematiksel yazılım	Bilgisayara kodlanması
$a + b - c + 2abc - 7$	$a + b - c + 2 * a * b * c - 7$
$a + b^2 - c^3$	$a + b^2 - c^3$
$a - \frac{b}{c} + 2ac - \frac{2}{a+b}$	$a - b/c + 2 * a * c - 2/(a+b)$
$\sqrt{a+b} - \frac{2ab}{b^2 - 4ac}$	$(a+b)^{(1/2)} - 2 * a * b / (b^2 - 4 * a * c)$
$\frac{a+b-c}{\sqrt{a^2+b^3}} - \frac{2(ab+ac+bc)}{9}$	$(a+b-c)/((a^2+b^3)^{(1/2)}) - 2 * (a * b + a * c + b * c) / 9$
$\frac{a^2+b^2}{2ab}$	$(a^2+b^2)/(2 * a * b)$
$A + \frac{B.C}{D} - E.F$	$A + B * C / D - E * F$

$a + \sqrt[3]{abc} - \frac{1}{1 + \frac{1}{a + \frac{1}{b + \frac{1}{c + \frac{1}{abc}}}}}$	$a + (a * b * c)^{(1/7)} - 1 / (1 + 1 / (a + 1 / (b + 1 / (c + 1 / (a * b * c)))))$
$\sqrt[3]{\frac{\sqrt[3]{a-b}}{\sqrt[4]{a+b-\frac{c}{ab}}}} + \frac{1}{\sqrt{1 + \frac{1}{\sqrt{1 + \frac{1}{abc}}}}}$	$((a-b)^{(1/5)} / (a+b-c / (a * b)))^{(1/4)}^{(1/3)} + 1 / ((1 + 1 / ((1 + 1 / (a * b * c))^{(1/2)}))^{(1/2)})$

While the mathematical expressions are written, if the parentheses are not paid attention, the results will be wrong. Below is how the same expressions with different locations of their brackets change the result.

$$\begin{aligned}
c*d/(a*d)+b+c*d/a &\Rightarrow \frac{cd}{ad}+b+\frac{cd}{a}=\frac{c}{a}+b+\frac{cd}{a} \\
c*d/a*d+b+c*d/a &\Rightarrow c\frac{d}{a}d+b+\frac{cd}{a}=\frac{cd^2}{a}+b+\frac{cd}{a} \\
c*d/a*d+(b+c)*d/a &\Rightarrow c\frac{d}{a}d+\frac{(b+c)d}{a}=\frac{cd^2+(b+c)d}{a}
\end{aligned}$$

3.2. Desicion Processes

The computer language equivalents of comparison signs in mathematics are different. It is very important to pay attention to these provisions when writing programs on the computer. These are as follows.

İşlem sembolü	Anlamı
=	Eşittir
<>	Eşit değildir
>	Büyüktür
<	Küçüktür
>= veya =>	Büyük eşittir
<= veya =<	Küçük eşittir

3.3. Logical Processes

The logical operations are explained by AND, OR, and NOT statements. Their use is similar to our daily use.

Mantıksal işlem	Komut olarak
VE	AND
VEYA	OR
DEĞİL	NOT

Example:

$a > 10$ and $b > 10$	is BINGO!
$a > 10$ or $b > 10$	is YESS!
$a > 10$ not	is HEYY!

Accordingly, the results to be obtained for the following values of a and b:

for a = 12, b = 15	BINGO YESS!
for a=8 , b=15	YESS! HEYY!
for a=13, b=8	YESS!

4. VISUAL BASIC SCRIPT (VbScript)

The Visual Basic programming language is a language that is easy to understand and has commands close to the spoken language.

In order to write basic program codes, firstly how to do loop, comparison, screen printing and variable definition with VbScript and functions will be mentioned, then the objects used in VbScript will be detailed.

4.1. Response.Write():

It is a method to print data on the client's screen in VbScript. It means something like "Return and Print" to the client. It can be thought of as the Print command in VB. It corresponds to the "information-data writing" process in flow diagrams. The Write method writes a specified string to the output.

Usage:

A = "Karabük" Response.Write(A)	It prints Karabük on the browser screen. Because the value of Karabük is assigned to variable A in the upper line.
Response.Write("Karabük")	It prints Karabük on the browser screen. Because we define what it will print. (Pay attention to the quotation marks!)
Response.Write("Karabük" & "Safranbolu")	It prints KarabükSafranbolu on the browser screen. The & sign means write side by side. (For string expressions, + can be used instead of &.)
A="Karabük" Response.Write(A & "Safranbolu")	It prints KarabükSafranbolu on the browser screen. Because A is variable and has been determined in the upper line. Safranbolu is written in quotes, so it will be printed out.

Using the Response.write method, it is also possible to write HTML code with VbScript, that is, you can do some kind of trick:

Example:

```
Response.Write("<body bgcolor= '#000000'>")
```

When you write, the server will send <body bgcolor = '# 000000'> to the client. Since this expression is a HTML interpretable by the browser, the browser will interpret it and change the page color. An equal sign can also be used instead of Response.write, provided that it is used with the VbScript tag:

Ex1: <% = ("Karabük")%> Returns the same result as: Response.Write ("Karabük")

Ex2: <% = A%> Returns the same result as: Response.Write (A)

Attention: If an equal sign will be used instead of Response.Write, it must be used among VbScript tags as above. Even if the tag has been opened before, it should be closed and opened again with the equal sign.

4.2. For-Next Loop

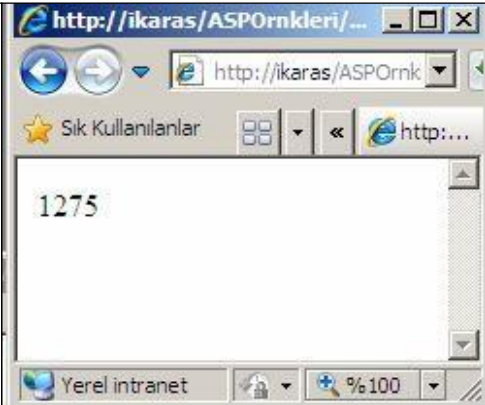
Coding of loop processes in flow diagrams is done with these commands in VbScript.

Usage:

FOR variable = initial value TO end value STEP quantity
..... NEXT

The NEXT statement here indicates the loop end and return to beginning point. In other words, the cycle that starts with FOR goes to NEXT and completes its first cycle. When it sees NEXT, it returns to FOR and starts the second cycle. In this way, the cycle continues until it is finished.

Example: A VbScript program code that calculates the sum of numbers from 1 to 50 and returns it to the client:

<pre><% Option Explicit Dim t, i t = 0 For i = 1 To 50 t = t + i Next Response.Write(t) %></pre>	 <p>The screenshot shows a web browser window with the address bar displaying 'http://ikaras/ASPOrnk...'. The main content area shows the number '1275'. The browser's status bar at the bottom indicates 'Yerel intranet' and '100%' zoom.</p>	<p>Numbers from 1 to 50 are summed and the result(1275) is sent to the browser client. The cycle will increase one by one as STEP is not written. When the value of a variable is 50, the program will cycle 50 times between FOR and NEXT.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Sample files: fornex1.asp, fornex2.asp <http://www.ismailkaras.com/asp/fornex1.asp>
<http://www.ismailkaras.com/asp/fornex2.asp>

Do While / Do Until Loop)

Its structure is different from ForNext. Exiting the loop is conditional. The cycle continues in two ways: 1-While the condition continues (while), 2- As long as the condition ends (until). The condition can be either at the beginning of the cycle or at the end.

Examples:

a=1 do while a<100 a=a+1 b=b+a Loop	a=1 Do a=a+1 b=b+a Loop while a<100
a=1 do until a=100 a=a+1 b=b+a Loop	a=1 Do a=a+1 b=b+a Loop until a=100

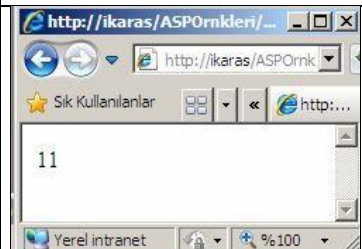
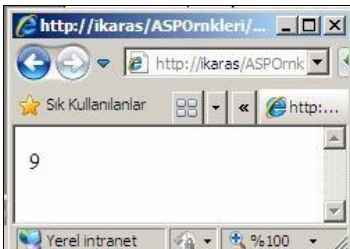
4.3. IF-THEN Statement

It is the equivalent of Decision-Comparison expressions in flow diagrams. The flow of the program varies depending on the criteria specified here.

Usage:

IF condition THEN1 END IF	When the condition is valid, the operations specified in the dotted section will be made.
IF condition THEN1 ELSE2 END IF	When the condition is valid, the procedures specified in section 1 will be done, and if not, in section 2 will be carried out.
IF condition1 THEN1 ELSEIF condition2 THEN2 ELSEIF condition3 THEN3 ELSE4 END IF	When Condition1 is valid, the operations specified in section 1 are performed. If not, new conditions can be added (condition2, condition3,...). The next condition is considered for each new condition that is not valid. If none of them are provided, the ones in section 4 will be made.

Example:

<pre><% Option Explicit Dim a, b, c a=5 b=6 a < 10 Then c = a + b Else c = a - b End If Response.Write(C) %></pre>	<pre><% Option Explicit Dim a, b, c a=15 b=6 a < 10 Then c = a + b Else c = a - b End If Response.Write(C) %></pre>	<p>If a value is less than 10, a and b will be summed, if it is large, it will be subtracted and the result will be sent to the client's browser.</p>
		

Exp:

```
<%
Option Explicit Dim a, b, c
a=7
b=6
```



```

If a < 10 Then
  c = a + b
elseif a>10 and a<13 then
  c = a * b
elseif a>=13 and a<17 then
  c = a / b
Else
  c = a - b
End If
Response.Write( C )
%>

```

If the condition can be expressed in a single line, it can be written without using "end if": IF condition THEN transaction

Exp:
 <% If a < 10 Then c = a + b %>

Sample files related to the subject: If1.asp, if2.asp <http://www.ismailkaras.com/asp/If1.asp>
<http://www.ismailkaras.com/asp/if2.asp>

Select-Case Statement

It is used to guide depending on the situation.

Exp:

```

Select Case Hour(Now)
  Case 0,1,2,3,4,5,6,7,8,9,10,11
    Response.Write "Good Morning!"
  Case 12,13,14,15,16,17
    Response.Write " Good Afternoon"
  Case Else
    Response.Write "Good Evening!"
End Select

```

4.4. Variables

As you can see in the flow diagrams and program examples, we often use variables when coding. Variables are elements that are temporarily filled in, allowing us to store the entered or generated data. It can be compared to terms like x, y in mathematics. Variables can consist of letters, numbers (numbers) and some signs, but some rules must be followed when naming variables. Variables must definitely begin with a letter, they cannot start with a number or a sign. Space or Turkish characters should not be used in the letters of the defined variables. Below are the names of variables defined as true and false.

A, R, c, s, x, pi, ali, isim, ISIM, no, ogrenci_no, a1, F4, memleketi	TRUE
1A, 6t, _rt, uyruğu, İSİM, adı soyadı, +,	WRONG

As you can see in the examples above, variables in VBScript are defined with the Dim command. If <% Option Explicit%> is used at the beginning of VbScript code, all variables must be defined by the user. If this statement is not used, the server will accept these variables without definition. However, it is better to use the expression <% Option Explicit%> to avoid misinterpretation of the server and then define all the variables ourselves.

ARRAYS

Variables can hold a single value, or they can hold multiple values when defined as an array. Arrays in VBScript are equivalent to matrices in mathematics. Just like in matrices, nxm-sized arrays can be created and elements can be assigned to that array by the number of elements in this dimension.

EXAMPLE 1 (This example shows assigning random numbers between 0 and 1 to a 4x3 array and printing these numbers on the client's screen. See the Functions section for the random number generation function):

<http://www.ismailkaras.com/asp/dizi.asp>

```
<%  
Dim A(3, 2)  
For i = 0 To 3  
    For t = 0 To 2  
        randomize  
        A(i, t) =rnd()  
        Response.write ( "A(" & i & "," & t & ") = " & A(i, t))  
        Response.write ("<br>")  
    Next  
Next  
%>
```

Although the examples below are VB 6.0 codes, when you examine them carefully, you will understand how to input and read data into the arrays.

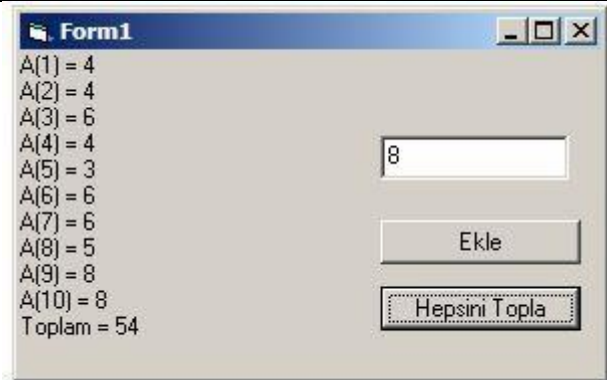
Example 2:

```
Dim t As Integer
Dim a(10) As Integer

Private Sub Command1_Click()
t = t + 1
a(t) = Text1.Text
Print "A(" & t & ") = " & a(t)
End Sub

Private Sub Command2_Click()
For p = 1 To t
    tp = tp + a(p)
Next p
Print "Toplam = " & tp
End Sub

Private Sub Form_Load()
t = 0
End Sub
```

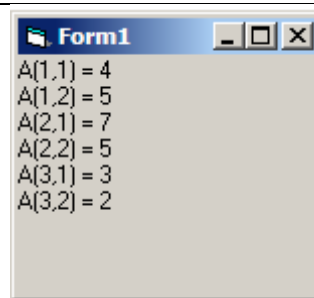


As defined at the top left, the a sequence is a 10x1 array. Each time the Add button is pressed, the value written in the text box is added to the array and written to the form. When the Collect All button is pressed, all elements of the series are summed with each other and written to the end.

Example 3:

```
Dim A(3, 2) As Integer

Private Sub Form_Load()
For i = 1 To 3
    For t = 1 To 2
        A(i, t) = InputBox("A(" & i & ", " & t & ") değeri giriniz...")
        Print "A(" & i & ", " & t & ") = " & A(i, t)
    Next t
Next i
End Sub
```



As defined in the top left, the A sequence is a 3x2 array. With the program code on the side, first the element is entered into this series by using the InputBox function, and then the elements are listed on the form as above.

(For other examples with arrays, see: gunler.asp)

4.5. ASCII Code Table

Each of the basic characters (letters, numbers, symbols and signs) used in the computer has a code. This code is as listed below. These codes are frequently used during programming. Accordingly, the ASCII code of the letter H is 72, the code of the letter h is 104, the code of the 5 is 53, etc.

0		32		64	@	96	`	128	Ç	160	Á	192	À	224	α
1	☐	33	!	65	À	97	a	129	ü	161	Í	193	⌚	225	ß
2	☐	34	"	66	B	98	b	130	é	162	Ó	194	⌚	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	Ú	195	⌚	227	Π
4	♦	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	Σ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	†	229	σ
6	♣	38	&	70	F	102	f	134	ã	166	±	198	†	230	μ
7	•	39	'	71	G	103	g	135	ç	167	±	199		231	τ
8	☐	40	(72	H	104	h	136	ê	168	¿	200		232	ϑ
9	◊	41)	73	I	105	i	137	ë	169	⌚	201		233	θ
10	☐	42	*	74	J	106	j	138	è	170	⌚	202		234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203		235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204		236	ω
13	♂	45	_	77	M	109	m	141	ì	173	¡	205	=	237	ø
14	♂	46	.	78	N	110	n	142	Ä	174	«	206		238	€
15	✱	47	/	79	O	111	o	143	Å	175	»	207		239	Π
16	▶	48	0	80	P	112	p	144	É	176	☐	208		240	≡
17	◀	49	1	81	Q	113	q	145	æ	177	☐	209	⌚	241	±
18	±	50	2	82	R	114	r	146	ff	178	☐	210	⌚	242	≥
19	!!	51	3	83	S	115	s	147	ô	179		211	u	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	⌚	212	⌚	244	∫
21	§	53	5	85	U	117	u	149	ò	181	⌚	213	⌚	245	J
22	—	54	6	86	U	118	u	150	û	182		214	⌚	246	÷
23	±	55	7	87	W	119	w	151	ù	183	⌚	215		247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	⌚	216	⌚	248	•
25	↓	57	9	89	Y	121	y	153	ö	185	⌚	217	⌚	249	·
26	→	58	:	90	Z	122	z	154	Û	186		218	⌚	250	•
27	←	59	;	91	[123	{	155	Ç	187		219	☐	251	√
28	↵	60	<	92	\	124		156	£	188		220	☐	252	η
29	↵	61	=	93]	125	}	157	¥	189		221	☐	253	z
30	▲	62	>	94	^	126	~	158	℔	190	⌚	222	☐	254	■
31	▼	63	?	95	_	127	△	159	f	191	⌚	223	☐	255	

ASCII Table

4.6. FUNCTIONS IN VBSCRIPT

The use of functions in VbScript is similar to functions in mathematics. They process a variable or value in a certain way and return a new result. Their rule is generally the same; The function is written first, then the value / variable and criteria are written in parentheses. For example: SQR (A) or SQR (36). There are hundreds of different functions. Here we will see some basic functions and uses.

Some basic functions:

ASC Function

Returns the ASCII code of the first character of the given string.

Example use 1: ASC (B): Generates the ASCII equivalent of the first letter of the value contained in variable B.

Example usage 2: ASC ("Karabük"): Produces the ASCII equivalent of the letter K.

CHR Function

Returns the equivalent of an ASCII code given in characters.

Example Usage: CHR (112): It produces 112 character code, that is the letter p.

SPACE Function

It is used to obtain the given number of spaces.

Example Usage: Space (8): Produces 8 space characters.

ARRAY Function

Creates an array based on the values given in parentheses. The number of elements and elements of the array is determined by parameters in parentheses.

EXP:

```
<%  
yonler=Array("kuzey","guney","dogu","bati") for i=0 to 3  
response.write yonler(i) & "<br>" next  
%>
```

Sonuç:

kuzey guney dogu bati

STRING Function

It is used to generate a certain number of characters with a specific ASCII code.

Example Usage 1: STRING (6, 146): Produces 6 of 146 coded characters.

Example Usage 2: STRING (6, "R"): Generates 6 R letters.

EXP:

```
<%  
response.write string(5, "!") prints "!!!!!" on the page  
%>
```

RIGHT Function

Separates as many characters as specified from the right in a given variable or value.

Example Usage 1: Right (A, 5): Takes 5 characters from the right of variable A

Example Usage 2: Right ("Ankara", 4): Takes the "kara" letters.

LEFT Function

Separates as many characters as specified from the left in a given variable or value.

Example Usage 1: Left (A, 5): Takes 5 characters from the left of variable A

Example Usage 2: Left("Ankara", 4) : Returns "Anka" letters.

EXP:

```
<%
str="Kuze eylen toprağım sunun anınla yare su" response.write left(str,10)
'sayfaya "Kuze eylen" yazar
%>

<%
str="Kuze eylen toprağım sunun anınla yare su" response.write right(str,7)
'sayfaya "yare su" yazar
%>
```

MID Function

Separates n characters from a mth character in a given variable or value.

Example Usage 1: Mid (A, m, n)

Example Usage 2: Mid (A, 5, 3): Separates 3 characters from the 5th character of the variable A.

Example Usage 3: Mid ("Ankara", 3, 2): Returns the letters "ka".

EXP:

```
<%
str="Kuze eylen toprağım sunun anınla yare su" response.write mid(str,6,5)
'prints "eylen" on the page.
%>
```

LEN Function

Returns the length, in characters, of a given variable or value.

Example Usage 1: Len (A): Returns the length of the variable A in characters.

Example Usage 2: Len ("Ankara"): Returns 6. Because Ankara is six characters.

EXP:

```
<%
str="Karabük Üniversitesi Mühendislik Fakültesi" response.write len(str)
'prints "42" on the page.
%>
```

LCASE Function

Converts a given variable or string to lowercase.

Example Usage 1: Lcase (F): Converts the value in variable F to lowercase.

Example Usage 2: Lcase ("ANKARA"): converts into "ankara"

UCASE Function

Converts a given variable or string to uppercase.

Example Usage 1: UCase (A): Converts the value in variable A to uppercase.

Example Usage 2: UCase ("ankara"): Converts into "ANKARA"

Note: It is necessary to create a special function for Turkish characters.

EXP:

```
<%
response.write lcase("AsP") 'prints "asp" on the page.
response.write ucase("asP") 'prints "ASP" on the page.
response.write lcase("KiTAP") 'prints "kitap" on the page.
response.write ucase("tırnak") 'prints "TıRNAK" on the page.

%>
```

TRIM Function

Removes spaces on either side of a given variable or string.

Sample Usage : Trim(" ankara ")

LTRIM Function

Removes spaces on left side of a given variable or string.

RTRIM Fonksiyonu

Removes spaces on right side of a given variable or string.

ÖRN:

```
<%
str=" Fuzûli "
response.write Ltrim(str) 'prints "Fuzuli " on the page.
response.write Rtrim(str) 'prints " Fuzuli" on the page.
response.write trim(str) 'prints "Fuzuli" on the page.
%>
```

REPLACE Function:

Replaces an expression specified in a string with another expression.

Usage :

```
replace(string,find,replacewith,start,count,compare)
```

string: Required. The string to be searched

find: Required. The part of the string that will be replaced

replacewith: Required. The replacement substring

start: Optional. Specifies the start position. Default is 1. All characters before the start position will be removed.

count: _num: Optional. Specifies the number of substitutions to perform. Default value is -1, which means make all possible substitutions

compare: Optional. Specifies the string comparison to use. If you give 0, it differentiates between upper and lower case, if you give 1, it does not. Default is 0

```
<%
str="Karabük Üniversitesi Edebiyat Fakültesi"

response.write replace(str,"Edebiyat","Fen",1,-1,1)
'Prints "Karabük Üniversitesi Fen Fakültesi" on the page

response.write replace(str,"edebiyat","Mühendislik",1,-1,0)
'Prints "Karabük Üniversitesi Edebiyat Fakültesi" on the page
(pay attention to the comparison value!)
```

```
%>
```


INSTR Function:

It searches for the expression we give in a text (String) and returns it to us as a value if it finds it. If not, the return value is 0 (zero).

Usage :

```
instr(start,string1,string2,compare)
```

start: Optional. Specifies the starting position for each search. The search begins at the first character position (1) by default. This parameter is required if compare is specified

string1: Required. The string to be searched

string2: Required. The string expression to search for

compare: Optional. Specifies the string comparison to use. If you give 0, it differentiates between upper and lower case, if you give 1, it does not. Default is 0

```
<%
str="Karabük Üniversitesi Edebiyat Fakültesi"
response.write instr(1,str,"Üniver",1) 'prints 9

str="Karabük Üniversitesi Edebiyat Fakültesi"
response.write instr(1,str,"üniver",0) 'prints 0

(pay attention to the comparison value!)
%>
```

SPLIT Function:

Creates a one-dimensional array of variables using the separators in the variable we provide

Usage :

```
split(expression, delimiter, count, compare)
```

```
<%
str="Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği
Bölümü"

StrDizi=split(str,"i",-1,1)

for t=0 to ubound(strdizi)

response.write "strDizi(" & t & ") = " & strDizi(t) & "<br>"

next

response.write ubound(strdizi)

%>
```

LBOUND and UBOUND Functions:

Returns the smallest and largest index values of an array. The smallest index value is naturally 0.

Usage:

LBound(arrayname[,dimension])

UBound(arrayname[,dimension])

Parameters:

arrayname: required.

dimension: Opsiyonel. Optional. Which dimension's lower bound to return. 1 = first dimension, 2 = second dimension, and so on. Default is 1

Examples:

Example 1:

```
<%  
days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")  
response.write(LBound(days) & "<br />")  
response.write(UBound(days) & "<br />")  
%>
```

Result:

0

6

Example 2:

```
<%  
Dim food(2,3)  
food(0,0)="Apple"  
food(0,1)="Banana"  
food(0,2)="Orange"  
food(0,3)="Lemon"  
food(1,0)="Pizza"  
food(1,1)="Hamburger"  
food(1,2)="Spaghetti"  
food(1,3)="Meatloaf"  
food(2,0)="Cake"  
food(2,1)="Cookie"  
food(2,2)="Icecream"  
food(2,3)="Chocolate"  
response.write(LBound(food,1) & "<br />")  
response.write(UBound(food,1) & "<br />")  
response.write(LBound(food,2) & "<br />")  
response.write(UBound(food,2) & "<br />")  
%>
```

Result:

0

2

0

3

CCur() Function

CCur(Number)

The CCur function is used to convert any entered value into Currency. The number range that can be made is:

-922,337,203,685,477.5808 to 922,337,203,685,477.5807

Sample code:

```
<% orneksayi=(12345) %>
```

```
<% =CCur(orneksayi) %>
```

Output:

12345

This function rounds to 4 decimals.

Sample code:

```
<% orneksayi=(55555.123456) %>
```

```
<% =CCur(orneksayi) %>
```

Output:

55555.1235

Some Mathematical Functions:

SIN, COS and TAN Functions

They take the sine, cosine, or tangent of a given variable or value. The angle value should be in radians. (180 degrees = 3.1415 .. radians)

Sample usage : cos(C) , sin(3.14)

ATN Function

Gets the arctangent of a given variable or value.

LOG Function

Gets the logart of a given variable or value.

Sample Usage: log (C), log (54)

SQR Function

It takes the square root of a given variable or value.

Sample Usage: sqr(C) , sqr (36)

CINT, INT and FIX Functions

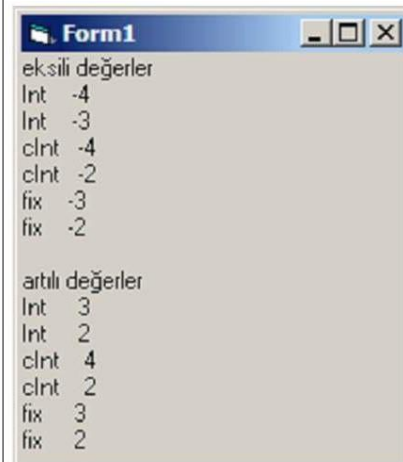
Returns the integer part of a given numerical variable or value, or rounds it up or down. This changes depending on the function selected.

ÖRNEK 4:

```
Private Sub Form_Click()

Print "eksili değerler"
Print "Int "; Int(-3.65)
Print "Int "; Int(-2.35)
Print "cInt "; CInt(-3.65)
Print "cInt "; CInt(-2.35)
Print "fix "; Fix(-3.65)
Print "fix "; Fix(-2.35)
Print: Print "artılı değerler"
Print "Int "; Int(3.65)
Print "Int "; Int(2.35)
Print "cInt "; CInt(3.65)
Print "cInt "; CInt(2.35)
Print "fix "; Fix(3.65)
Print "fix "; Fix(2.35)

End Sub
```



INT, CINT ve FIX fonksiyonlarının nega
vada varının üstünde veya altında olu

RND

It is used to generate random numbers.

Exp:

<%

```
dim a
randomize
a=rnd()
response.write a
```

%>

Time and Date Functions:

Almost all time-date functions of Visual Basic are also used in VBScript.

Date: Returns the date today. (Like 25.03.2000)

Time: Returns the current time. (Like 22:24:40)

Now: Gives the current date and time together. (Such as 25.03.2000 22:24:40)

VBScript also has Weekday (day of the week), WeekdayName (name of the day) and Monthname (name of the month) functions. These functions get their values according to the Date function. For example,

<%= WeekdayName(Weekday(Date))%>

command gives us the value of "Saturday" if it is Saturday.

```
<%= MonthName(Month(Date))%>
```

command gives us the value "March" if this month is March. In addition to these, Day, Month and Year functions of VBScript also take parameters from Date function and give you a number. If the date is March 25, 2000:

```
<%= Day(Date)%>
```

```
<%= Month(Date)%>
```

```
<%= Year(Date)%>
```

return value.

VBScript gets these values directly from the operating system. If the regional settings of the operating system is set to Turkey, the names of days will return as Turkish. In addition, the date and time formats also change according to regional settings, month ahead, day back, or vice versa, the clock on a 12-hour or 24-hour basis. While trying your ASP programs on personal Web Server, the date and time of your own computer; you get the date and time of the Server while running on the real Internet. In order to display the month and day names in Turkish on your pages, you may first need to try the server's regional settings and if the names do not come in Turkish, you may need to type subs or functions that convert them. (In our code that gives the date of the day above, regardless of the region settings of your system, the month will appear in Turkish.)

EXAMPLE:

```
<%= Date%><br>
```

```
<%= Day(date)%><br>
```

```
<%= Month(Date)%><br>
```

```
<%= Year(Date)%><br>
```

```
<%=Weekday(Date)%><br>
```

```
<%=Weekday("12.06.1967")%><br>
```

```
<%= WeekdayName(Weekday("12.06.1967"))%><br>
```

```
<%=Month(Date)%><br>
```

```
<%= MonthName(Month(Date))%>
```

Dynamic Array Function (Dynamic Array)

Array variables, especially in recording data from our Web visitors; It is a useful tool in making the data extracted from the database available.

Therefore, you will frequently use multi-dimensional array variables in your ASP pages. For this, it would be appropriate to briefly and collectively discuss the necessary tools. When creating a series of variables, if we don't specify the number of elements of the variable, VBScript says to itself, "They want me to make this array dynamic!" he says. It then creates a dynamic array variable whose values can be specified later and the number of elements can be increased later. Sample:

```
Dim Students ()
```

With this command, the Students array variable is created. However, since the number of elements is not specified, the array is dynamic. You can then determine the number of elements in this array. This,

ReDim Ogrenciler(15)

We can do it with a command like. Now the question may come to your mind: “Why don't we define the series of students by specifying the number of staff from the beginning?” Nice question! Could the answer be the following? “We don't yet know the number of elements of our array variable. In the flow of the program, this number can be determined as the result of another function, sub or user input.” But there is one thing to note right away: The ReDim command deletes everything inside an existing set of variables! If we want to protect the elements of the current array and their values:

ReDim Preserve Ogrenciler(20)

we need to write. The Preserve command here tells VBScript to protect the elements in the current array and increase the number of elements to 20. Why might this be needed? The visitor's preferences may change. For example, your visitor can buy new things on an electronic shopping site. You need to increase the number of elements of the array variable that you hold the data for your previous shopping without deleting the previous information.

You can think of VBScript's array variables as a list all with the same name; there is only a number next to the variable name indicating the number of elements of the array:

Ogrenciler(1): Necip
Ogrenciler(2): Serap
Ogrenciler(3): Neslihan

But VBScript can also create multi-dimensional array variables. Think of the two-dimensional array variable as a table. The elements of the array are variables of the same name, but this time they are determined not only by odd numbers but by row and column numbers:

Ogrenciler(1,1): Necip
Ogrenciler(1,2): Serap
Ogrenciler(1,3): Neslihan
Ogrenciler(2,1): Selim
Ogrenciler(2,2): Murat
Ogrenciler(2,3): Merve
Ogrenciler(3,1): Elif
Ogrenciler(3,2): Hande
Ogrenciler(3,3): Leyla

Now, you can bring a three-row, three-column table here. We can create this array-variable with the command:

Dim Ogrenciler(3,3)

In such a variable, for example, the first row (those with the number 1, x) can indicate hardworkers, the second row (2, x's) can specify less hardworkers. VBScript creates three, four, and even five-dimensional array variables. You can decide where to use it.

You can change the value of any element of a set of variables at any stage of the program:

```
Ogrenciler(3,2) = "Caner"
```

The command deletes the name “Hande” and replaces the name “Caner in its place.

We may want to know the number of elements of our array variables. Sometimes we do not determine the number of elements of our array variables. This information may come from a form; can be taken from a database. However, for example, it is necessary to know how many elements this variable for a loop. If we have 35 element array, we have this number,

```
NumberOfElement = UBound (Ogrenciler)
```

With the command, we print the NumberOfElement variable. The number of NumberOfElement will be 35 in this case.

Examples on the topic: [dynamic_array_.htm](#) >> [dynamic_array_.asp](#)

Test Functions

The current status of some of the variables we use in VBScript can provide the information we will use to control the flow of our program. For example, if the value of a variable is empty, we may think that our visitor does not fill out the form completely. VBScript provides some special functions for us to test the state of variables. The value returned from these special functions is True or False; the value of the correct result is -1, and the value of the false result is 0:

IsArray:	Tests whether a variable is an array variable.
IsDate:	Tests whether a variable's value can be converted to date.
IsEmpty:	Tests whether a variable has been defined and assigned a value.
IsNull:	Tests whether a variable holds a valid value.
IsNumeric:	Tests whether a variable can be treated as a number.
IsObject:	Tests whether an expression references a valid ActiveX or OLE object.
TypeName:	Specifies the type of a variable.
VarType:	Returns the number that specifies the type of a variable.

Example with the Functions

<p>Example 1:</p> <pre><% Dim a Dim d a = "ankara anafartalar caddesi" d = Len(a) For p = 1 To d If Mid(a, p, 1) = "a" Then Response.write (p) End If Next p >%</pre>	<p>1 4 6 8 10 12 15 17</p> <p>When the code is executed, letter a positions in the sentence "ankara anafartalar caddesi" will be listed. In this example, you see MID and LEN functions, variable naming, FOR-NEXT loop, IF-THEN conditional statements, Response.write command.</p>
<p>Example 2:</p> <pre><% a = " Karabük " b = "Safranbolu" Response.write (b & a & b & "
") Response.write (b & Trim(a) & b & "
") Response.write (b & LTrim(a) & b & "
") Response.write (b & RTrim(a) & b & "
") >%</pre>	<p>KarabükSafranbolu KarabükSafranbolu Karabük Safranbolu Karabük Safranbolu</p> <p>Using TRIM, LTRIM and RTRIM functions</p> <p>The word Karabük is first written without trimming the spaces to its left and right. Then the spaces on both sides of the bottom line are cropped and Safranbolu ends. Then it is written by cropping from the left and right.</p>

<p>Example 3: http://www.ismailkaras.com/asp/int_cint_fix.asp</p> <pre> <% Response.write ("eksili değerler" & ""
"") Response.write ("Int " & Int(-3.65) & ""
"") Response.write ("Int " & Int(-2.35) & "
") Response.write ("cInt " & CInt(-3.65) & "
") Response.write ("cInt " & CInt(-2.35) & "
") Response.write ("fix " & Fix(-3.65) & "
") Response.write ("fix " & Fix(-2.35) & "
") Response.write ("artılı değerler" & "
") Response.write ("Int " & Int(3.65) & "
") Response.write ("Int " & Int(2.35) & "
") Response.write ("cInt " & CInt(3.65) & "
") Response.write ("cInt " & CInt(2.35) & "
") Response.write ("fix " & Fix(3.65) & "
") Response.write ("fix " & Fix(2.35) & "
") %> </pre>	<p>Negative Values</p> <pre> Int -4 Int -3 cInt -4 cInt -2 fix -3 fix -2 </pre> <p>Positive Values</p> <pre> Int 3 Int 2 cInt 4 cInt 2 fix 3 fix 2 </pre> <p>INT, CINT and FIX change of functions according to whether they are negative, positive or above or below half.</p>
<p>Example 4:</p> <pre> <% a = "FEN EDEBİYAT FAKÜLTESİ" Response.write (UCase(Left(a, 1))); LCase(Right(a, Len(a) - 1)) %> </pre>	<p>Fen edebiyat fakültesi</p> <p>In this example usages of UCASE, LCASE, RIGHT, LEFT and LEN functions are shown</p>

EXAMPLE:

Other sample files related to the functions are below. Try these files and examine the codes:

Garfield.asp, Tesadüfi.asp, Dogumgunu.asp, yazi-tura.asp, ornek_string_fonk.asp, ornek_chr_fonk.asp, ornek_split_fonk.asp, ornek_split_fonk_.asp, ornek_tarih_saat_fonk.asp, ornek_test_fonks.asp, hosgeldiniz01.asp, hosgeldiniz02.asp, dogumgunu01.asp, dogumgunu02.asp, Trk_krkt_bul_buyut.asp, split_ubound.asp, int_cint_fix.asp

EXAMPLE (saat.asp):

Conversion server clock time in the United States to Turkey (TR 10 hours ahead) and AM / PM system to 24-hour system conversion process. Examine the example, consider all the alternative.

```

Current time in US: <%=time()%>
<%
'Change the current USA time to local time:
sat = split(time(),"",-1,1)
if right(time(),2)="PM" and sat(0)<>12 then
    sat(0) = sat(0) + 12
end if
sat(0) = sat(0) + 10
if sat(0)>=24 then
    sat(0)=sat(0)-24
end if
saat_tr = sat(0) & ":" & sat(1) & ":" & left(sat(2),2)
%>
<br>
Current time in Turkey: <%=saat_tr%>

```

PROCEDURE AND USER DEFINED FUNCTIONS

Procedures (sub) and user-defined functions are similar. They both work like a subprogram. When writing code, we use the procedure if we want the program to go elsewhere and complete the operations there. The difference of user-defined functions from the procedure; sending to the subprogram with one or more data, processing that data and returning. In this way, it is possible to generate new functions similar to the ready functions of VBScript.

EXAMPLE (procedure.asp):

```

<% dim f,t, c, p, r Hesap
f = t + c + p + r
response.write f

Sub Hesap
t = 3
c = (Sin(t) + 5) * Log(t)
p = Tan(c)
r = Cos(p)
end sub %>

```

(Detailed example of the procedures can be examined in the example named `giris_yap_.asp` on the following pages.)

EXAMPLE (function.asp):

<% 'In this example, instead of using the `Sqr()` function, which is the ready function of VBScript, we produce our own square root function:

```

response.write karekok(49)

```

```
Function Karekok(x)
Karekok = x ^ (1 / 2)
end function %>
```

EXAMPLE:

Mail Control with User Defined Function:

While entering the information of our visitors, we may not check whether the e-mail addresses they provide are real e-mail addresses, but at least we can check whether the e-mail address they provide is logically correct. First of all we have to establish the logic of the event.! Only one "@" character and at least one "." in an actual mail address. (dot), no prohibited characters, and most importantly, an email address can be a minimum of "6" characters. Here is our function ...

```
<%
Function MailKontrol(StrMail) 'value to our function StrMail

MailKontrol = False
'Let's give "False" value to our function

Yasak = array("/", "\", "(", ")", "[", "]", "{", "}", "*", "?")
'We have identified the forbidden characters and we will use them later
'You can make additions if you want

if len(StrMail) < 6 then
MailKontrol = False
'the total number of characters is less than 6 ...
exit function 'no need to continue, exit
end if

At = 0 '@
Nokta = 0

For i = 1 to len(StrMail) 'we are returning the number of characters
karakter = mid(strMail,i,1)
'we take the characters one by one

if karakter = "@" then
'if the current character is equal to "@"
At = At + 1
'It is incremented by 1
end if

if karakter = "." then
'maybe it is a dot
Nokta = Nokta + 1
end if

'we loop in the loop
'we compare the current character with invalid characters!
```

```

for j = Lbound(Yasak) to Ubound(Yasak)
'It is cycled from index to end of array variable
if karakter = yasak(j) then
'we compare the character we have with the invalid characters in the loop
MailKontrol = False
'We made the value of our function False
exit function
'There is no need to continue; exit
end if
next
'we are finishing the inner loop

next
'Let's not mix it for this next outer loop' checks begin

if At = 1 and Nokta>=1 then
'@ character is only 1 and the number of dots is at least 1
MailKontrol = True
'the information given to us is correct
else
'if not means wrong
MailKontrol = False
end if

end function
%>

<%
gelen_adres = request.form("gelen_adres")
'we take the variable from the form
if MailKontrol(gelen_adres) then
'we send it to our function
he gives the necessary answer to the other side
response.write "Thank you, registration is ok"
else
response.write " I think there is a problem with your Mail Address"
end if
%>

```

(Please look at the similar example, see: email.asp)

FORMS

(Read this section with Forms section in HTML Presentation-6 document.)

In web pages, we all use objects such as textbox, button (button) for data entry. For example, when logging into a site, we enter our username and password in two different textboxes and click the send button. Objects such as textbox, button, listbox, checkbox are part of the forms. A web page containing form objects does not have to contain any script code. Form pages can be designed with HTML only. However, the information sent by the form must be evaluated and interpreted by a server-side program. So even though the forms issue seems to be a part of HTML, we will cover the topic here under the heading of ASP.

Like all HTML tags, forms begin and end with a tag. The start tag is <FORM> and the end tag is </FORM>. All processes and objects related to the form are written between these two labels.

Sections of the Forms

The HTML form has three parts. These are the section that shows what the action the Web designer expects from the form and addresses the visitor's browser; sections that the visitor should fill in or options to choose; and command buttons that allow the visitor to activate or cancel the action command of this form.

Action ve Method

The <FORM> tag that you will use to create a form in your web area also includes instructions to the browser program of the user on what to do with the information in this form. For this, in the FORM tag, if this form is filled in and sent by ACTION to the browser, you tell where, at which address, to which program the information will be delivered. Since the HTTP protocol allows two types of communication between the Web Server and the client's computer, in this section, you should also tell the browser which method to choose with the METHOD attribute.

Therefore, rule of the Form label will be:

<FORM ACTION="url" METHOD=POST>

Here, instead of url letters, the address of the program that will process the data that will come with this form will be found. For example: "gonder.asp"

Gaps and Markings to be Filled

You need to fill in between the <FORM> ... </FORM> tag with either spaces that the user will fill in, or lists and buttons that allow him to choose.

Your main controllers providing this are INPUT, SELECT and TEXTAREA tags. Let's examine them in order:

INPUT METHOD

With the INPUT tag, we give the client the opportunity to enter information by marking the form with the keyboard or using the mouse. The general way to use this tag is as follows:

<INPUT TYPE="..." NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..." SCR="..." CHECKED" ">

Now, let's examine the usage principles of this tag according to the work that the user can do:

TEXTBOX

- **To enter text with the user's keyboard:**

TYPE=TEXT NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..."

The digit "Size" determines the width in which this box will be displayed on the user's screen as characters; The digit "Maxlength" specifies the length of the text that the user can enter. If you do not put this digit and do not give a value, the browser assumes the maximum text length as 21 characters. When this box is shown on the screen, if you want to have a text inside it, write it in quotes to the value "Value = ...". For the form to reach the server and be recognized when interpreted by an ASP page, you must name it by typing it in the "Name = ..." field.

PASSWORD TEXTBOX

- **To enter password with the user's keyboard:**

TYPE=PASSWORD NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..."

It has the same features as the textbox; however, the information that the user will enter in this box is not displayed on the screen, but the star icon is displayed.

HIDDEN

- **To send confidential data from the page:**

type=hidden name=login value=true

No data is entered from the screen, the data is sent hiddenly in the background. For example, the user can name his name in the relevant boxes, etc. When the button enters the button, the system time can be sent along with the information entered. In this case, the clock is sent as hidden. For example;

<input type=hidden name="saat" value=<%=time()%> >

CHECKBOX

- **To mark a box**

TYPE=CHECKBOX NAME="..." VALUE="..." [CHECKED]

The client can enter a "x" or remove it automatically by clicking the mouse in the checkbox you will create with this command or by pressing the spacebar on the keyboard. If there is a check in this box, the browser sends the information you will write to the value "Value =" and the name of the box to the Server. If the box is unchecked, the name and value of the box will not be sent to the Server. When creating this box, you must enter a name in the Name field in quotes; otherwise, the information that is coming may not be useful for you. If you want the box to be checked automatically, include the word CHECKED; If you don't want to, don't write this word. Write what this box does before or after creating this box.

OPTIONBOX

- **To put a black dot in a round space (Radio button):**

TYPE=RADIO NAME="..." VALUE="..." [CHECKED]

All features and principles of this element are like CHECKBOX, except that the place to be marked is a circle rather than a square box.

It is also possible to put graphic or hidden text on the form with the INPUT tag. Instead of the BUTTON tag, which creates a button that the user can click with the mouse icon on the screen that comes with HTML, operations that enable the user to send the information to the Server, such as Enter and Delete, or to empty the information it entered completely.

SELECT (LISTBOX)

Using this tag, you can create a box in the form and a down arrow next to it; By clicking on the down arrow, the user can select an item from the list to open and have it written in the box. If you wish, one of these elements can also be displayed automatically selected. The use of this label is as follows:

<SELECT NAME="...." SIZE="..." [MULTIPLE]>..... </SELECT>

You can determine how many options the box on the screen will show by typing 1, 2, or 3 etc .. If this digit is not set, the automatic 1 option is assumed. The options to be shown in this box are written between the <SELECT> .. </SELECT> tags with the <OPTION> tag. (The <OPTION> tag is not closed.) To automatically select any option, the word SELECTED is put inside the <OPTION> tag. Sample:

```
<select name = "Contact" size = "1"> <option selected> Please make a choice <option  
value = "Phone"> Phone <option value = "Email"> Email <option value = "Come">  
Personal Interview </ select> </ p>
```

Example:

```
<select name="Temas" size="1">  
<option selected>Lütfen bir tercih yapınız  
<option value="Telefon">Telefon  
<option value="EPosta">E Posta  
<option value="Gel">Şahsi Görüşme  
</select>
```

TEXTAREA

Textarea allows to send much longer text than textbox towards the sever. The use of this label is as follows: `<TEXTAREA NAME="..." rows=.. cols=..>` The text that is automatically written in the box is written here. `</TEXTAREA>`

With the "Name =" attribute, the necessary name can be given to the text box for processing and using the text to be sent to the Server. The numbers to be given against the "rows=" and "cols=" indicate how many lines of height and letter width of this box will take place in the form. These two measures don't affect the length of the text to be entered in the box.

Send and Delete Buttons

A Submit button must be placed at the end of the form logically, enabling the user to send the information that he / she has filled and the preferences he made to the Internet area that presents the form. And depending on the preference, a Delete button can be placed so that those who want to change the information they write on the form and their preferences can be deleted completely.

For this, the INPUT tag is used as follows:

```
<INPUT TYPE=SUBMIT NAME=Gonder VALUE="Gönder">
<INPUT TYPE=RESET NAME=Sil VALUE="Sil">
```

EXAMPLE:

Run the ExampleForm.htm file (with Babyweb) and examine the results in the FormOrnek.asp file.

REQUEST

Data sent with a form like above is obtained with the request.form method. Exp:

Sending: `<input type="text" name="kullanici" size="20">`

Receiving: `<%Request.form("kullanici")%>`

Note: Request.form can only be used as Request. Exp: Request("kullanici")

If the information is sent with a link, not with the form, it is obtained with the Request.querystring method.

Exp:

Sending: `İlk Sayfa`

Receiving: `<%Request.QueryString("Sayfa")%>`

EXAMPLE (Request.Form):

try and examine the files gonderdim.htm> aldim.asp

EXAMPLE (Request.QueryString):

1. gonderdim_qs.htm > aldim_qs.asp
2. linkleverigonder.htm > FormOrnek.asp
3. Try and examine the files named cikansec.asp in the QString folder (It should be in the same directory with STOK.mdb).

In this file, you can see the topics such as sending static and dynamic data, paging, and progress between pages with the link. Column headers send data statically and page navigation links dynamically. Pay attention to the calculation of the number of pages depending on the number of records to be displayed on a page. Consider that the number of records is exact value (eg 3000). If ks number of records, the number of pages:

$$\text{sonsayfa} = (\text{int}((\text{ks}-1)/500)) + 1$$

DATA STORING METHODS

The history of the filing method, which is sometimes used today in the storage of data in computer environment, computers are old. Although so-called "database" systems have been developed that perform this process in much superior conditions, filing method can still be preferred in some small applications due to its simple structure and direct access. What are the differences and advantages of these two methods?

1.1. Classic Method: File Operations

In the filing process, the data is stored in one or more files, directly in the recording medium, and processing (recording, query, updating, deletion) is carried out on them through application programs. In the organization of the data by the filing method, each application program directly accesses the data files. While preparing application programs, all kinds of activities in the recording environment, from the way of recording the data to the place, should be considered, kept under control and designed accordingly. These programs have to know how to store data in files. That is, application programs must contain all the necessary commands for accessing the data file. This situation causes many repetitions. When any changes are made to the data files, the commands that provide access should also be updated separately in each application program. [1] Another big problem, security, arises when data files are shared by different users and different application programs in a network environment. The control and safety of the data is not fully provided or requires a lot of effort. Which user is able to access data, which is authorized to change ... these must be thought out and planned very well and must be specified one by one in the application programs.

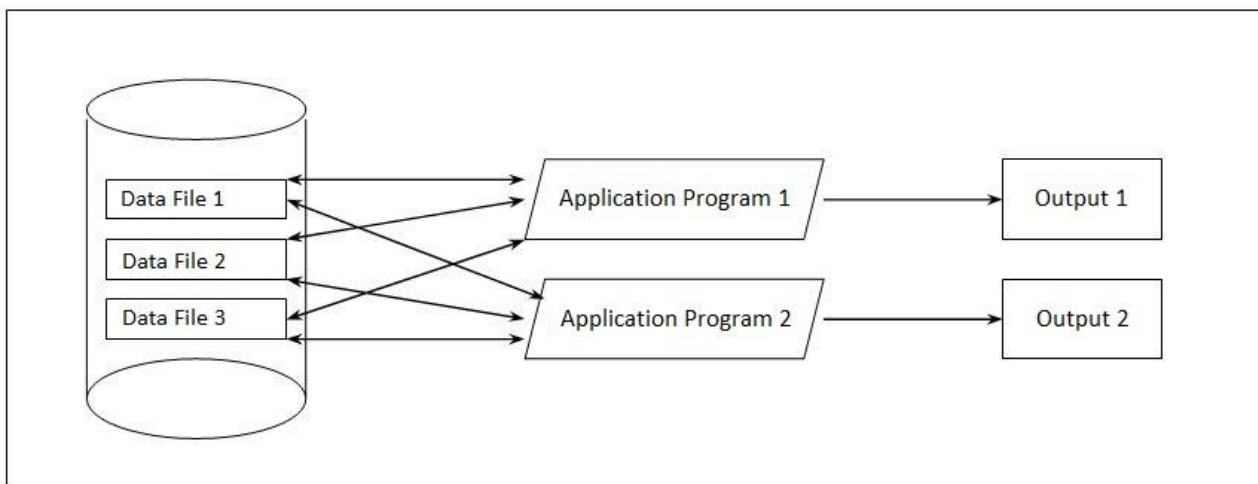


Figure 1.1. Data and application program relation according to filing method.

EXAMPLE:

See the examples below for the filing method:

konuk01.htm, konuk02.htm, konuk_isle.asp: data entry to text file

konuk_oku.asp: read data in text file

1.2. Database Concept

“The database is a software system that allows the data to be stored, updated and accessed with special techniques, which are free from unnecessary repetitions, provided with accuracy, consistency, privacy and security, by means of special languages that the user can easily learn in order to use them in many independent applications. (Fischer, 1993) ”[2]

The concept of database is a concept that has been reached after long experience and stages in the world of computing, and as an alternative to classical file management, it is the result of the development of software that can accommodate large capacity, fast, large data stacks, and software that meets the demands of the comprehensive, network environment. It has increased. The most important feature in a classic filing system is to be application dependent; that is, whatever software a file was created by, depending on the software, the file can be accessed; however, in database management, there is principal data-application independence; In other words, it is possible to access the data created once with any programming language or application program. [*] [3]

The database system consists of a database and special software that manages it. This special software is called database management software / system (DBMS). Software such as Access, Dbase, Oracle, Paradox are such software. The database is a collection of interrelated data and keeps not only data, but also the relationships between them. The structures of modern databases based on the relational model used today are similar. [3] The data are recorded in tables consisting of rows (records) and columns (fields), and different tables containing common or related data can be associated. “Database management software, recording, deleting, editing, querying, indexing, multi-user reading, updating, sharing etc. It performs operations like, organizes and checks data access routes, authorizations and data integrity. Apart from these, the main program related user interfaces, forms, menus, reports, queries, macros, etc. is one of the services provided by database management software. With these services they provide, today's database management software can fulfill the tasks of application programs in many respects, and special purpose software can be added thanks to macros. (Healey, 1993) ”[2]

One of the important benefits of database management software is that it provides data independence. Application programs do not engage in where and how the data is saved, it only transmits the relevant request, the database software performs the desired process in the background, and even more quickly than filing. This software acts as both a tool and a controller between data and application programs. Any changes to the database or application programs do not affect another, however, database management software ensures that the data will be provided correctly. Thus, separate workload is not required to protect the application programs and database. The services provided by the database system also facilitated the development of new application programs. [1]

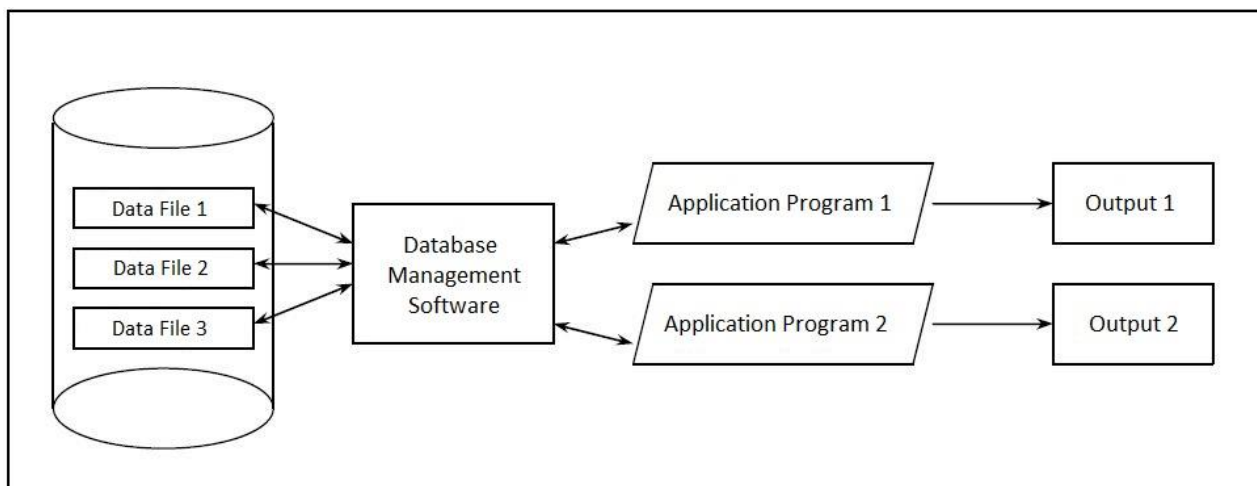


Figure 1.2. The relationship between data files, database management system and applications

1.2.1. Advantages of the Database

The advantages of the database over the filing method can be summarized as follows;

1. Database management software serving one or more users ensures that the data standards are determined and presented in a single center, the security conditions are met, the incompatibilities are eliminated, are ensured the integrity of the database. Thanks to this central control, user authorizations are determined and constantly monitored.
2. Through management software, a single database can be used simultaneously by different users, in different applications, and can be shared safely and quickly.
3. Application programs operate independently of the physical environment where data is stored. Application programs do not need to know the data structure, since data access is via database management software.
4. New application programs and database applications can be easily integrated into the system through the services provided by the database management software.
5. In the filing process, different data files are used for each application, resulting in significant data excess. There may be valid reasons for storing multiple copies of some data. However, excessive data repetition is a waste of labor, time and cost. An effective database management software ensures that data is properly stored as well as updated copies.
6. The database management software is easy to use through the menus, queries, reports and interfaces it provides. Functions such as backup and monitoring can restore the deletions or unwanted changes occurring in the data.

1.2.2. Disadvantages of the Database

Database systems may have some disadvantages besides their benefits. These can be listed as follows:

1. Software and hardware of database systems can have high costs. However, thanks to appropriate and efficient implementation programs, the benefits obtained in the long term will exceed this.
2. Database systems are more complicated than filing. In theory, complex systems are negatively affected by operations on data.
3. Theoretically, during data transfers while the application program is running, there is a large risk of data being lost or corrupted. However, backup and correction procedures are often provided by the database management software and this risk is minimized.

Data Record Into the Database With VBScript

Sample code for entering new records in the "IsimSoyad" (NameSurname), "Yasi" (age), and "KayitTarihi" (RecordDate) fields in the table "BenimTablom1" belonging to the database "Veritabanim.mdb":

```
'Setting up DB connections:
Set Baglantim = CreateObject("ADODB.Connection")
'Opening DB:
Baglantim.Open ("DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" &
Server.MapPath("Veritabanim.mdb"))
'Creating the table object:
Set Tablom = server. CreateObject("ADODB.Recordset")
'Opening the table:
Tablom.Open "BenimTablom1", Baglantim, 1, 3

'Start adding data to the table:
Tablom.AddNew
'Transfer data to fields in table
Tablom("IsimSoyad") = request("AdiSoyadi")
Tablom("Yasi") = request("Yas")
Tablom("KayitTarihi") = request("KayitTr")
'Update of the table after the transfer process:
Tablom.Update

'Closing the table:
Tablom.close
set Tablom= Nothing
'Closing the connection:
Baglantim.close
set Baglantim= Nothing
```

EXAMPLE (Samples in the VTUyg folder):

Try and examine the files named VeriGirisi.asp and VeriGirisiOK.asp (Must be in the same folder with Veritabanim.mdb.)

Reading Data from Database with VBScript

Sample code to search for a specific record in the "IsimSoyad" field in the table "BenimTablom1", which belongs to the database "Veritabanim.mdb":

```
<%
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("Veritabanim.mdb"))
ssql="select * from Benimtablom1 ORDER BY IsimSoyad;"
Set oRS = oConn.Execute(ssql)
```

Do While NOT oRS.EOF

```
if oRS("IsimSoyad") = Request.form("AdSoyad") then
%>
<%=oRS("IsimSoyad")%> <br>
<%=oRS("Yasi")%><br>
<%=oRS("KayitTarihi")%><br>
<%
end if
oRS.MoveNext
Loop

oConn.Close
Set oRS = Nothing
Set oConn = Nothing %>
```

EXAMPLE:

arama_TextBox.asp > BulGetir.asp, arama_tablosuz.asp > BulGetirtablosuz.asp,
arama_Combolu.asp > BulGetir.asp
examine and try these files (with Veritabanim.mdb)

Listing All Records in the Database with VBScript

Sample code to list all records in the database named "BenimTablom1", belonging to the database "Veritabanim.mdb":

```
<%
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("Veritabanim.mdb"))
ssql="select * from BenimTablom1; "
Set oRS = oConn.Execute(ssQL)
```

Do While NOT oRS.EOF

```
%>
<%=oRS("IsimSoyad")%>
<%=oRS("Yasi")%>
<%=oRS("KayitTarihi")%>
<br>
<%
oRS.MoveNext
Loop
```

```
oConn.Close
Set oRS = Nothing Set oConn = Nothing
%>
```

EXAMPLE:

VeriListeleme_tablolu.asp
VeriListeleme_tablosuz.asp
examine and try these files (with Veritabanim.mdb)

Updating a Record in the Database with VBScript

Sample code to fix a specific record in the database named "BenimTablom1", belonging to the database named "Veritabanim.mdb":

```
<%  
dsn = "DBQ=" & Server.MapPath("veritabanim.mdb") & ";Driver={Microsoft Access Driver  
(*.*.mdb)}";  
Set conn=Server.CreateObject("ADODB.Connection")  
conn.Open dsn
```

```
SQL = "Update BenimTablom1 Set Yasi = '" & Request.Form("Yas") & "', KayitTarihi  
= '" & Request.Form("KayitTr") & "' Where IsimSoyad = '" & request.form("AdiSoyadi") & "'"
```

```
Set RS = conn.Execute(SQL)  
conn.Close  
Set conn = Nothing  
>
```

EXAMPLE:

Arama_Duzeltme.asp > VeriDuzeltme.asp > VeriDuzeltmeOK.asp ListedenDuzeltme.asp >
VeriDuzeltme.asp > VeriDuzeltmeOK.asp
Try and examine the files named (with veritabanim.mdb)

Deleting a Record in the Database with VBScript

Sample code to delete a specific record in the table "BenimTablom1" belonging to the database named "Veritabanim.mdb":

```
<%  
Set oConn = Server.CreateObject("ADODB.Connection") oConn.Open("DRIVER={Microsoft  
Access Driver (*.mdb)}; DBQ=" & Server.MapPath("veritabanim.mdb"))
```

```
set kayit_sil = Server.CreateObject("ADODB.RecordSet")
```

```
SQL = "delete * from BenimTablom1 Where IsimSoyad = '" & request.form("AdiSoyadi") & "'"
```

```
kayit_sil.Open sql, oConn, 1, 2
```

```
set kayit_sil=nothing  
oConn.CLOSE  
SET oConn = NOTHING  
>
```

EXAMPLE:

Arama_Silme.asp > VeriSilme.asp > VeriSilmeOK.asp
ListedenSilme.asp > ListedenSilmeOK.asp
Try and examine the files named (with veritabanim.mdb)

APPLICATION and SESSION OBJECT

In terms of ASP, a site is considered as an "**application**". Each visitor is determined as a "**session**". What is the reason why we call the Site, which consists of a set of ASP and HTML pages, is called a **session** for any visit? We can explain this with the functions of both objects.

The **Application** object keeps information about the entire site (variables, objects and methods); The **session** object keeps track of the visitor's entrance to our site.

Sometimes we want that the value of a variable be the same on all pages; the value of the variable does not change for the visitor when changing the page. It is very easy to do with ASP. In order to overcome this difficulty in ASP, we can create our variables for the **Session** object; and this value continues throughout the visitor's session; can be known by all Functions on all ASP pages.

For example:

```
Session ("Tupras") = 44500
```

it creates a valid Tupras variable for the entire Session and assigns it the value "44500". Sometimes we may want the variable to be much more extensive, that is, its lifetime is not specific to Session but throughout the entire Application. Then we can define this variable at the Application level:

```
Application ("Tupras") = 44500
```

In this case, the Tupras variable will have the same value for all visitors.

In order for the **session** object to occur, we must send a **cookie** to the visitor and give a sign on our site. Whereas, if we give the visitor a **Session ID** as soon as it connects to our site and check this ID every time when a new page request, we know who is maintaining the session. An ASP-compatible Web Server keeps every **Session** object open for 20 minutes, unless the visitor makes a new choice; then deletes. You can change this time through the **Timeout** property of the **Session** object. The **session cookie** is automatically sent and tracked by the ASP-compatible Web Server; As a designer, we don't have to do anything about it.

How many people access your Web site at the same time (i.e. how many people request your pages), so many **Session** objects are created; but since your site has one, there is one **Application** object. There is a file that allows this object to ensure the needs of our site for all Session and have the same rules of practice: **Global.asa**. This file is created when installing PWS or IIS (must be created in the home directory). In this file, it is often written that **Application_OnStart**, which occurs with the entry of the first visitor to our site, and **Application_OnEnd**, which occurs with the appearance of the last visitor, and **Session_OnStart**, which occurs when a visitor accesses a page, and **Session_OnEnd** events that occur when the visitor leaves our site. Although the content of this file is similar to a standard ASP file, the reason of **.asa** extension instead of **.asp** is being **Active Server Application** file. An ASP-compatible Web Server program runs the **Global.asa** file as soon as it detects the first visitor to our site.

Where **Application** and **Session** objects are used the most is to keep the number of visitors (Hits received by our site) on our site. This is usually done by placing a counter in the **Global.asa** program.

EXAMPLES (In the session directory):

LOGIN1 Example:

1. For login and control operations by looking at user names and passwords in the database;

Try the files named `Giris.htm` > `login.asp` > `index.asp` and examine their codes (All of them should be in the same folder as `BenimVT.mdb`).

LOGIN2 Example:

1. For login and control processes;

Try the files named `giris_yap .asp` > `menu.asp` and examine the codes.

"Uname" variable is moved to the `menu.asp` file with the Session ("uname") object. The links in `menu.asp` are shown whether this variable is full or not. When the session is ended, the variable will not hold any value and then the links will not be shown to the user and will be redirected to the login page.

2. For login and control operations by looking at user names and passwords in the database;

`giris_yap_.asp` > `menu.asp` Try and examine the files (Should be in the same directory with `Bolum.mdb`)

3. Determining the session duration;

Examine the codes of the file named `global_time.asa`. The session time in the file is set to 1 min. To see the sample, after logging in by entering http://www.ismailkaras.com/giris_yap.asp (username: 466 password: bilgisayar), wait more than a minute on the menu page and refresh the page. As the session expires (because the `uname` variable is "nothing"), the menus will not be shown and will be redirected to the login page. (Change the file name `global_time.asa` to `global.asa` when trying. `Global.asa` must be in root directory, not subdirectories)

4. To see how many clients are connected to the site;

Try the files named `global_sayac.asa` and `visitor.asp` and examine their code (Change the file name `global_sayac.asa` to `global.asa`. `Global.asa` must be in root directory, not subdirectories)

RESPONSE OBJECT

Buffer;

As soon as ASP sees the response object, it sends the result of the current operation directly to the user. If you are going to make a decision about the view of the page as a result of an operation to be done in the later stages of the page, you can make this value true and send it to the user at the end of the page. Unless it is changed, the value of the buffer property is false. General use is

```
<%response.buffer=[true/False]%>
```

If it gets True value, it stores the page in memory and sends it at the end.

If it gets False value, it sends the response directly to the user.

Expires;

Some pages are stored in user browsers to be opened quickly in their next use. With Expires value, you can determine how much time (in minutes) will be stored in the cache of this page in the user browser. If the value is zero or minus, the page is never stored in cache. The usage;

```
<% Response.expires=[number] %>
```

Write;

You can print everything on the page with response.write. This method is the most used method of ASP.

```
<% response.write [data to be printed] %>
```

Example: <% response.write "Computer Engineering" %>

Redirect;

This function provides to go directly to a page;

Example: <% response.redirect "menu.asp" %> The page will automatically

redirect to the menu.asp page.

INCLUDE FILE

This method is used to show any file / content on multiple web pages. Usage:

```
<!-- #include file="altmenu.htm" -->
```

ERROR OBJECT

It is an inevitable rule to make mistakes when writing a program in any language. Therefore, the programming language you use should allow you to easily catch your mistakes. In the ASP programs, there should be no script error such as spelling error, sending to non-variable. Such errors must be debugged before the program is sent to the Web. But there may be error situations that you cannot predict as a programmer, and that are caused by most Web visitors or the visitor's computer. VBScript can keep the program running in unexpected error situations with the following standard command:

```
<% On Error Resume Next %>
```

With this command, VBScript will continue from the next line in case of an error. However, the resulting error can prevent the program from delivering the expected result. VBScript can tell you through the properties of Err (Error) Object what the error is and where it originated, especially the error number (description) and source (Source) properties. Using these features, in your programs, in the development phase, for example,

```
If Err:Number = xx Then
```

According to the type of error, you can ensure that the program runs without any problem. You can make 108 different error numbers instead of xx. Error numbers are available in Microsoft's VBScript site.