



4th Class

Intelligent Applications

التطبيقات الذكية

أستاذ المادة : أ.م.د. حسنين سمير عبد الله

1. Introduction to Expert Systems

Expert systems are computer programs that are constructed to do the kinds of activities that human experts can do such as design, compose, plan, diagnose, interpret, summarize, audit, give advice. The work such a system is concerned with is typically a task from the worlds of business or engineering/science or government.

Expert system programs are usually set up to operate in a manner that will be perceived as intelligent: that is, as if there were a human expert on the other side of the video terminal.

A characteristic body of programming techniques give these programs their power. Expert systems generally use automated reasoning and the so-called weak methods, such as search or heuristics, to do their work. These techniques are quite distinct from the well-articulated algorithms and crisp mathematical procedures more traditional programming.

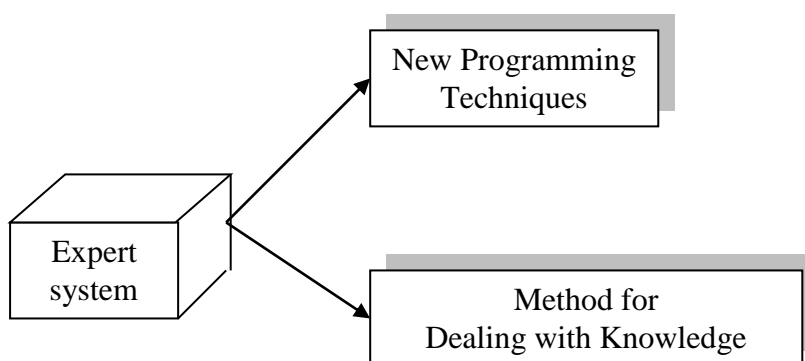


Figure (1) the vectors of expert system development

As shown in Figure(1), the development of expert systems is based on two distinct, yet complementary, vectors:

- a. New programming technologies that allow us to deal with knowledge and inference with ease.
- b. New design and development methodologies that allow us to effectively use these technologies to deal with complex problems.

The successful development of expert systems relies on a well-balanced approach to these two vectors.

2. Expert System Using

Here is a short nonexhaustive list of some of the things expert systems have been used for:

- To approve loan applications, evaluate insurance risks, and evaluate investment opportunities for the financial community.
- To help chemists find the proper sequence of reactions to create new molecules.
- To configure the hardware and software in a computer to match the unique arrangements specified by individual customers.
- To diagnose and locate faults in a telephone network from tests and trouble reports.
- To identify and correct malfunctions in locomotives.

- To help geologists interpret the data from instrumentation at the drill tip during oil well drilling.
- To help physicians diagnose and treat related groups of diseases, such as infections of the blood or the different kinds of cancers.
- To help navies interpret hydrophone data from arrays of microphones on the ocean floor that are used for the surveillance of ships in the vicinity.
- To figure out a chemical compound's molecular structure from experiments with mass spectral data and nuclear magnetic resonance.
- To examine and summarize volumes of rapidly changing data that are generated too fast for human scrutiny, such as telemetry data from landsat satellites.

Most of these applications could have been done in more traditional ways as well as through an expert system, but in all these cases there were advantages to casting them in the expert system mold.

In some cases, this strategy made the program more human oriented. In others, it allowed the program to make better judgments.

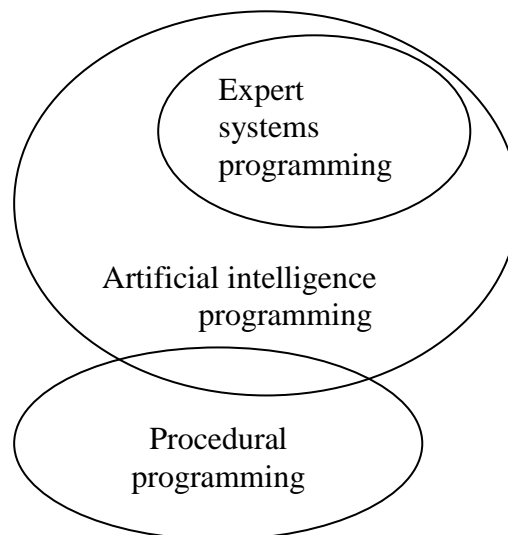
In others, using an expert system made the program easier to maintain and upgrade.

3. Expert Systems are Kind of AI Programs

Expert systems occupy a narrow but very important corner of the entire programming establishment. As part of saying what they are, we

need to describe their place within the surrounding framework of established programming systems.

Figure(2) shows the three programming styles that will most concern us. Expert systems are part of a larger unit we might call AI (artificial intelligence) programming. Procedural programming is what everyone learns when they first begin to use BASIC or PASCAL or FORTRAN. Procedural programming and A.I programming are quite different in what they try to do and how they try to do it.



Figure(2) three kinds of programming

In traditional programming (procedural programming), the computer has to be told in great detail exactly what to do and how to do it. This style has been very successful for problems that are well defined. They usually are found in data processing or in engineering or scientific work.

AI programming sometimes seems to have been defined by default, as anything that goes beyond what is easy to do in traditional procedural

programs, but there are common elements in most AI programs. What characterizes these kinds of programs is that they deal with complex problems that are often poorly understood, for which there is no crisp algorithmic solution, and that can benefit from some sort of symbolic reasoning.

There are substantial differences in the internal mechanisms of the computer languages used for these two sorts of problems. Procedural programming focuses on the use of the assignment statement (" = " or ":-") for moving data to and from fixed, prearranged, named locations in memory. These named locations are the program variables. It also depends on a characteristic group of control constructs that tell the computer what to do. Control gets done by using

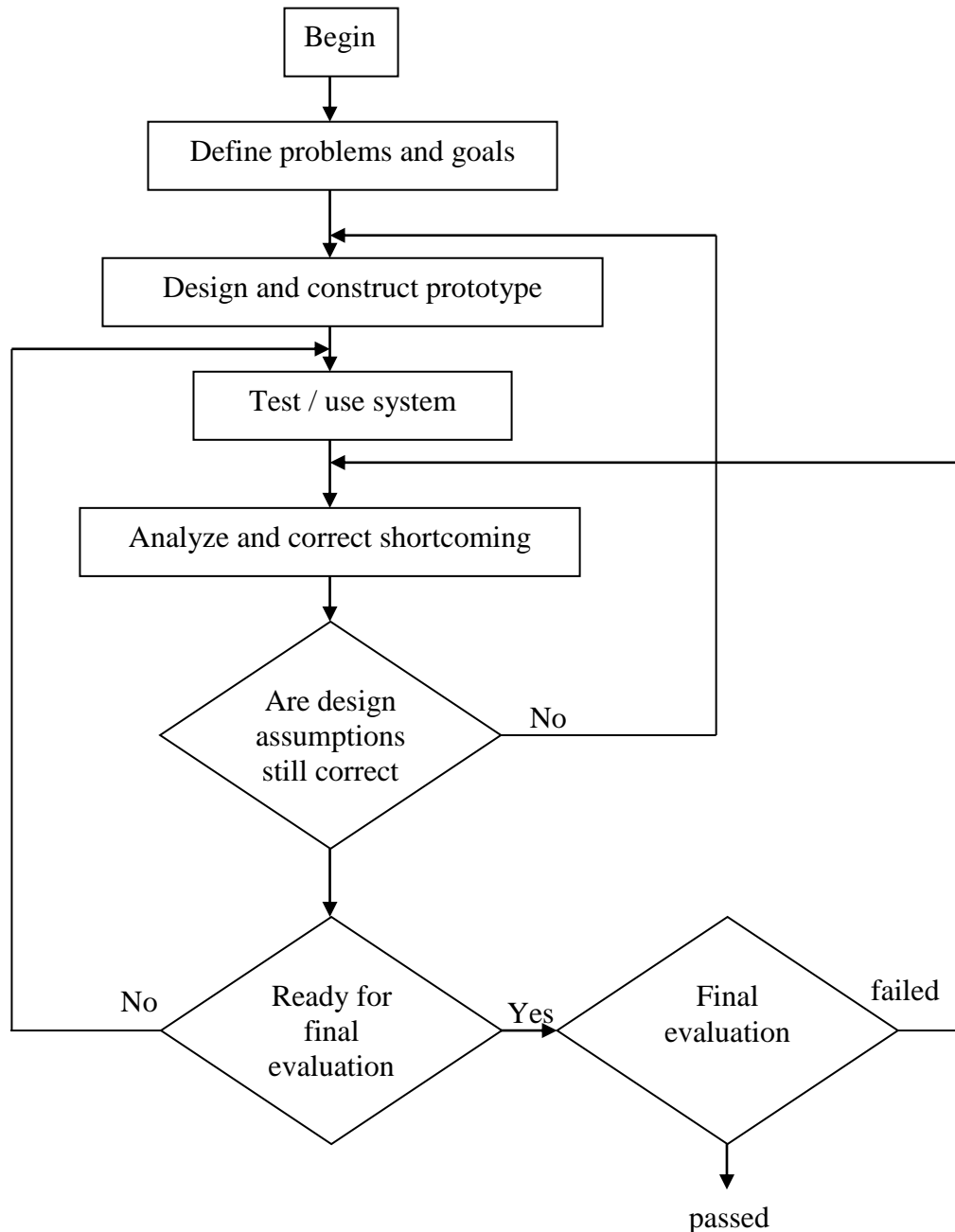
if-then-else	goto
do-while	procedure calls
repeat-until	sequential execution (as default)

AI programs are usually written in languages like Lisp and Prolog. Program variables in these languages have an ephemeral existence on the stack of the underlying computer rather than in fixed memory locations. Data manipulation is done through pattern matching and list building. The list techniques are deceptively simple, but almost any data structure can be built upon this foundation. Many examples of list building will be seen later when we begin to use Prolog. AI programs also use a different set of control constructs. They are :

- procedure calls
- sequential execution
- recursion

4. Expert System, Development Cycle

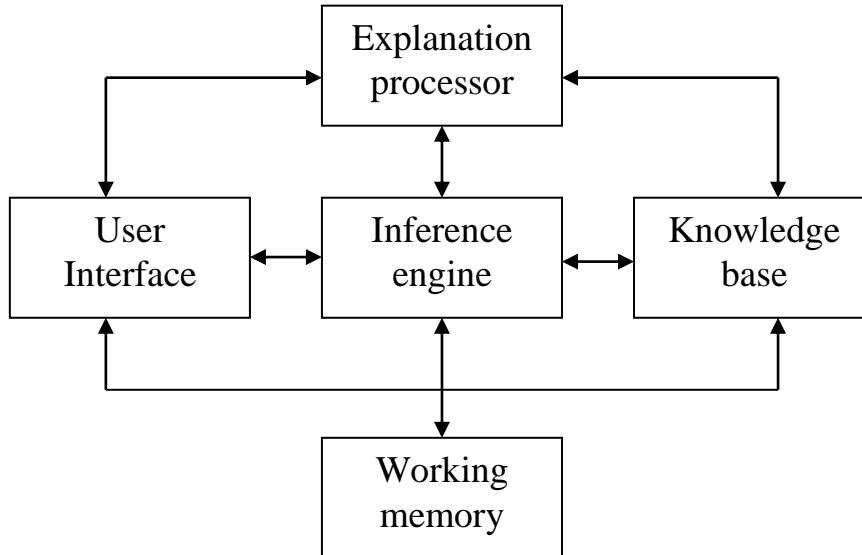
The explanation mechanism allows the program to explain its reasoning to the user, these explanations include justification for the system's conclusions, explanation of why the system needs a particular piece of data. Why questions and How questions. Figure (3) below shows the exploratory cycle for rule based expert system.



Figure(3) The exploratory cycle for expert system

5. Expert System Architecture and Components

The architecture of the expert system consists of several components as shown in figure (4) below:



Figure(4)Expert system architecture

5.1. User Interface

The user interacts with the expert system through a user interface that make access more comfortable for the human and hides much of the system complexity. The interface styles includes questions and answers, menu-driver, natural languages, or graphics interfaces.

5.2. Explanation Processor

The explanation part allows the program to explain its reasoning to the user. These explanations include justifications for the system's conclusion (HOW queries), explanation of why the system needs a particular piece of data (WHY queries).

5.3. Knowledge Base

The heart of the expert system contains the problem solving knowledge (which defined as an original collection of processed information) of the particular applications, this knowledge is represented in several ways such as if-then rules form.

5.4 Inference Engine

The inference engine applies the knowledge to the solution of actual problems. It s the interpreter for the knowledge base. The inference engine performs the recognize act control cycle.

The inference engine consists of the following components:-

1. Rule interpreter.
2. Scheduler
3. HOW process
4. WHY process
5. knowledge base interface.

5.5. Working Memory

It is a part of memory used for matching rules and calculation. When the work is finished this memory will be raised.

6. Systems that Explain their Actions

An interface system that can explain its behavior on demand will seem much more believable and intelligent to its users. In general, there are two things a user might want to know about what the system is doing. When the system asks for a piece of evidence, the user might want to ask,

"Why do you want it?"

When the system states a conclusion, the user will frequently want to ask,

"How did you arrive at that conclusion?"

This section explores simple mechanisms that accommodate both kinds of questioning. HOW and WHY questions are different in several rather obvious ways that affect how they can be handled in an automatic reasoning program. There are certain natural places where these questions are asked, and they are at opposite ends of the inference tree. It is appropriate to let the user ask a WHY question when the system is working with implications at the bottom of the tree; that is: when it will be necessary to ask the user to supply data.

The system never needs to ask for additional information when it is working in the upper parts of the tree. These nodes represent conclusions that the system has figured out, rather than asked for, so a WHY question is not pertinent.

To be able to make the conclusions at the top of the tree, however, is the purpose for which all the reasoning is being done. The system is trying to deduce information about these conclusions. It is appropriate to ask a HOW question when the system reports the results of its reasoning about such nodes.

There is also a difference in timing of the questions. WHY questions will be asked early on and then at unpredictable points all throughout the reasoning. The system asks for information when it discovers that it needs it. The time for the HOW questions usually comes at the end when all the reasoning is complete and the system is reporting its results.

domains

rxnlist = reactions*.

reactions = rxn(symbol, ls, integer, integer).

ls = symbol*.

chemicalList= chemicalForm*.

chemicalForm= chemical(symbol, rxnList, integer, integer).

Li= integer*.

predicates

rxn(symbol, ls, integer, integer).

rawmaterial(symbol, integer, integer).

chemical(symbol, rxnlist, integer, integer).

all_chemical(symbol, chemicalList).

best_chemical(symbol, chemicalForm).

one_chemical(symbol, chemicalForm).

append(rxnlist, rxnlist, rxnlist).

min(chemicalList, chemicalForm).

run(symbol).

clauses

rxn(a, [b1, c1], 12, 60).

rxn(b1, [d1, e1], 5, 45).

rxn(c1, [f1, g1], 3, 15).

rxn(a, [b2, c2], 10, 50).

rxn(b2, [d2, e2], 2, 20).

rxn(c2, [f2, g2], 6, 30).

rawmaterial(d1, 2, 0).

rawmaterial(e1, 0, 0).

rawmaterial(f1, 2, 0).

rawmaterial(g1, 0, 0).

rawmaterial(d2, 0, 0).

rawmaterial(e2, 1, 0).

rawmaterial(f2, 1, 0).

rawmaterial(g2, 0, 0).

chemical(Y, [], Cost, Time):- rawmaterial(Y, Cost, Time).

chemical(Y, L, Ct, T):-

rxn(Y, [X1, X2], C, T1), chemical(X1, L1, C1, T2), chemical(X2, L2, C2, T3),

append(L1, L2, Q), Ct = C+C1+C2,

T = T+T2+T3, append([rxn(Y, [X1, X2], C, T1)], Q, L).

best_chemical(Y, M):- all_chemical(Y, X), min(X, M).

all_chemical(Y, X):- findall(S, one_chemical(Y, S), X).

one_chemical(Y, chemical(Y, L, Ct, T)):- chemical(Y, L, Ct, T).

append([], L, L):-!.

append([H|T], L, [H|T1]):- append(T, L, T1).

min([chemical(Y, L, Ct, T)], chemical(Y, L, Ct, T)).

min([chemical(Y, L, Ct, Time) | T], chemical(Y, L, Ct, Time)):-

min(T, chemical(Y1, L1, C1, Time1)), Ct <= C1.

min([chemical(Y, L, Ct, Time) | T], chemical(Y, L2, Ct2, Time2)):-

min(T, chemical(Y, L2, Ct2, Time2)), Ct2 <= Ct.

run(X):- write(" chemical synthesis is:"), nl, chemical(X, L, Cost, Time),

write(L, "\n with total cost =", Cost, " Time =", Time), nl, fail.

run(X):- write("\n Best chemical synthesis:"), nl, best_chemical(X, Y), write(Y), nl.

Goal: run(a).

chemical synthesis:

[rxn("a", ["b1", "c1"], 12, 60), rxn("b1", ["d1", "e1"], 5, 45), rxn("c1", ["f1", "g1"], 3, 15)]

with total cost = 24 time = 120

[rxn("a", ["b2", "c2"], 10, 50), rxn("b2", ["d2", "e2"], 2, 20), rxn("c2", ["f2", "g2"], 6, 30)]

with total cost = 20 time = 100

best chemical synthesis :

chemical("a", [rxn("a", ["b2", "c2"], 10, 50) rxn("b2", ["d2", "e2"], 2, 20), rxn("c2", ["f2", "g2"], 6, 30)], 20, 100)

Controlling the Reasoning Strategy

Classification Program with Backward Chaining (Bird, Beast, Fish) Version1

database

db_confirm(symbol, symbol)

db_denied(symbol, symbol)

clauses

guess_animal :- identify(X), write("Your animal is a(n) ",X),!.

identify(giraffe) :-

 it_is(ungulate),
 confirm(has, long_neck),
 confirm(has, long_legs),
 confirm(has, dark_spots)

identify(zebra) :-

 it_is(ungulate),
 confirm(has, black_strips),!.

identify(cheetah) :-

 it_is(mammal),
 it-is(carnivorous),
 confirm(has, tawny_color),
 confirm(has, black_spots),!.

identify(tiger) :-

it_is(mammal),
it-is(carnivorous),
confirm(has, tawny_color),
confirm(has, black_strips),!.

identify(eagle) :-

it_is(bird),
confirm(does, fly),
it-is(carnivorous),
confirm(has, use_as_national_symbol),!.

identify(ostrich) :-

it_is(bird),
not(confirm(does, fly)),
confirm(has, long_neck),
confirm(has, long_legs),!.

identify(penguin) :-

it_is(bird),
not(confirm(does, fly)),
confirm(does, swim),
confirm(has, black_and_white_color),!.

identify(blue_whale) :-

it_is(mammal),
not(it-is(carnivorous)),
confirm(does, swim),
confirm(has, huge_size),!

identify(octopus) :-

not(it_is(mammal),
it_is(carnivorous),
confirm(does, swim),
confirm(has, tentacles),!

identify(sardine) :-

it_is(fish),
confirm(has, small_size),
confirm(has, use_in_sandwiches),!

identify(unknown). **/* Catch-all rule if nothing else works. */**

it-is(bird):-

confirm(has, feathers),
confirm(does, lay_eggs),!

it-is(fish):-

confirm(does, swim),
confirm(has, fins),!

it-is(mammal):-

confirm(has, hair),!

it-is(mammal):-

confirm(does, give_milk),!

it-is(ungulate):-

it-is(mammal),

confirm(has, hooves),

confirm(does, chew_cud),!

it-is(carnivorous):-

confirm(has, pointed_teeth),!

it-is(carnivorous):-

confirm(does, eat_meat),!

confirm(X,Y):- db_confirm(X,Y),!

confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).

denied(X,Y):- db-denied(X,Y),!

Check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y, Reply).

remember(X, Y, yes):- asserta(db_confirm(X, Y)).

remember(X, y, no):- assereta(db_denied(X, Y)), fail.

Controlling the Reasoning Strategy

Classification Program with Forward Chaining (Bird, Beast, Fish) Version2

database

db_confirm(symbol, symbol)

db_denied(symbol, symbol)

clauses

guess_animal :-

find_animal, have_found(X),

write("Your animal is a(n) ",X),nl,!.
.

find_animal:- test1(X), test2(X,Y), test3(X,Y,Z), test4(X,Y,Z,_),!.
.

Find_animal.
.

test1(m):- it_is(mammal),!.
.

test1(n).
.

test2(m,c):- it_is(carnivorous),!.
.

test2(m,n).
.

test2(n,w):- confirm(does, swim),!.
.

test2(n,n).
.

```
test3(m,c,s):- confirm(has, strips),
                asserta(have_found(tiger)),!.
test3(m,c,n):- asserta(have_found(cheetah)),!.
test3(m,n,l):- not(confirm(does, swim)),
                not(confirm(does, fly)),!.
test3(m,n,n):- asserta(have_found(blue_whale)),!.
test3(n,n,f):- confirm(does, fly),
                asserta(have_found(eagle)),!.
test3(n,n,n):- asserta(have_found(ostrich)),!.
test3(n,w,t):- confirm(has, tentacles),
                asserta(have_found(octopus)),!.
test3(n,w,n).

test4(m,n,l,s):- confirm(has, strips),
                 asserta(have_found(zebra)),!.
test4(m,n,l,n):- asserta(have_found(giraffe)),!.
test4(n,w,n,f):- confirm(has, feathers),
                 asserta(have_found(penguin)),!.
test4(n,w,n,n):- asserta(have_found(sardine)),!.

it-is(bird):- confirm(has, feathers),
              confirm(does, lay_eggs),!.
```

```
it-is(fish):- confirm(does, swim),  
            confirm(has, fins),!.
```

```
it-is(mammal):- confirm(has, hair),!.
```

```
it-is(mammal):- confirm(does, give_milk),!.
```

```
it-is(ungulate):- it-is(mammal),  
                confirm(has, hooves),  
                confirm(does, chew_cud),!.
```

```
it-is(carnivorous):- confirm(has, pointed_teeth),!.
```

```
it-is(carnivorous):- confirm(does, eat_meat),!.
```

```
confirm(X,Y):- db_confirm(X,Y),!.
```

```
confirm(X,Y):- not(denied(X,Y)),!, check(X,Y).
```

```
denied(X,Y):- db-denied(X,Y),!.
```

```
Check(X,Y):- write(X, " it ", Y, \ "n"), readln(Reply), remember(X, Y, Reply).
```

```
remember(X, Y, yes):- asserta(db_confirm(X, Y)).
```

```
remember(X, y, no):- assereta(db_denied(X, Y)), fail.
```

Conclusions

1. Code written for backward chaining is clearer. All the rules in version 1 of BBF have a nice declarative reading. They correspond nicely to most people's intuitive idea of how things should be described when they are part of some kind of hierarchy. The description is top down.
2. Code written for backward reasoning is also much easier to modify or expand. It is apparent without much thought what would have to be done to add another animal (class) to the structure: just define it. But it is not always clear where to attach another instance to a forward reasoning rule structure. In fact, if a number of additions have to be made, all the rules may have to be redone to accommodate the additions and at the same time to maintain the same testing efficiency as was there before.
3. Code for the backward reasoning system will be easier to develop in the first place because the built-in inference method in prolog is backward chaining.

Study Question

1. Show what would be required to add these two animals to both versions of BBF:
 - The camel, an ungulate with a hump.
 - The unicorn, an ungulate with a single horn.
2. By examining the listings of the BBF programs, calculate the average number of questions that will be asked to identify an animal in the forward chaining versions and in the backward chaining version.
3. Find a set of rules that describe what to do when your computer will not started. Organize the appropriate rules into both a backward chaining and forward chaining systems (version 1 & 2).

Programs that Work under Uncertainty factor

Approximation Reasoning and Bipolar States

Logical Implications

- Simple Implication

$$ct(c) = ct(e) * ct(imp)$$

- AND Implication

$$ct(c) = \min(ct(e1), ct(e2)) * ct(imp)$$

- OR Implication

$$ct(c) = \max(ct(e1), ct(e2)) * ct(imp)$$

Bipolar Calculation Values

- If $ct1(c)$ is +ve and $ct2(c)$ is +ve (+ +) then

$$Ct(c) = (ct1(c) + ct2(c)) - (ct1(c) * ct2(c))$$

- If $ct1(c)$ is -ve and $ct2(c)$ is -ve (- -) then

$$Ct(c) = (ct1(c) + ct2(c)) + (ct1(c) * ct2(c))$$

- If [$ct1(c)$ is +ve and $ct2(c)$ is -ve (+ -)] or

[$ct1(c)$ is -ve and $ct2(c)$ is +ve (- +)] then

$$Ct(c) = (ct1(c) + ct2(c)) / (1 - \min(\text{abs}(ct1(c)), \text{abs}(ct2(c))))$$

Reversible and non reversible Rules

Reversible

- If $ct(c)$ is -ve and prefaced by not then $Ct(c)$ is +ve
- If $ct(c)$ is +ve and prefaced by not then $Ct(c)$ is -ve

Non reversible

- If ct(c) is -ve and prefaced by not then Ct(c) is +ve
- If ct(c) is +ve and prefaced by not then Ct(c) = 1- (+ve)

Knowledge Base

- hypothesis_node(C).
- terminal_node(e).
- imp(logic op, rule type, conclusion name, left condition sign, left condition name, right condition sign, right condition name, imp value)

Systems that Explain their Actions

The HOW & WHY Facilities

Consider the following Inference Network (fuzzy net)

If e1 and e2 then c1 (imp= 0.8) rev

If not(e3) or c3 then c2 (imp= 0.9) rev

If e4 and e5 then c3 (imp= 0.8) rev

If c1 or c2 then c4 (imp= 0.8) rev

Answering WHY Questions

S: Type w(why) or give the certainty for node e4

U: w

S: Attempting to establish c3 via the implication

e4 and e5 \rightarrow c3

Type w(why) or give the certainty for node e4

U: w

S: Attempting to establish c2 via the implication

not e3 or c3 \rightarrow c2

Type w(why) or give the certainty for node e4

U: w

S: Attempting to establish c4 via the implication

c1 or c2 \rightarrow c4

Type w(why) or give the certainty for node e4

U: 0.85

(Why Stack Description)

Why stack description is explained through the lecture.

Answering HOW Question

S: Type h(how) nodename, or c (to continue).

U: h c4

S: Concluded c4 with certainty of 0.68 from

c1 or c2 \rightarrow c4

The rule is reversible

Certainty of c1 is 0.6

Certainty of c2 is 0.85

The certainty of the implication is 0.8

Used alone the rule suggests a certainty of 0.68

Type h(how) nodename, or c (to continue).

U: h c1

S: Concluded c1 with certainty of 0.6 from

$e1 \text{ or } e2 \rightarrow c1$

The rule is reversible

Certainty of e1 is 0.75

Certainty of e2 is 0.65

The certainty of the implication is 0.8

Used alone the rule suggests a certainty of 0.6

Type h(how) nodename, or c (to continue).

Consider the following Production rules:

if not(e3) or e4 then c1 (imp = 1.0) nrev

if not(e1) and not(e2) then c2 (imp = 0.8) rev

if c1 or e5 then c3 (imp = 0.7) nrev

if not(e6) then c4 (imp = 0.9) nrev

if e7 and e8 then c5 (imp = 0.8) nrev

if not(e9) then c5 (imp = 0.9) rev

if c2 then c6 (imp = 0.9) rev

if c3 then c6 (imp = 0.9) nrev

if c4 and c5 then c6 (imp = 0.85) nrev

e1= 0.2 e2= -0.2 e3= -0.2 e4= 0.7 e5= -0.5 e6= -0.8 e7= 0.8

e8= 0.8 e9= -0.7

Systems That Depend on Reasoning under Uncertainty

Approximate Reasoning (Structure of the FUZZYNET Program)

driver:- hypothesis-node(X), allinfer(X, Ct),

write("The certainty for ", X, "is", Ct), nl, fail.

allinfer(Node, Ct):- findall(C1, infer(Node, C1), Ctlist), supercombine(Ctlist, Ct).

/*A simple implication */

infer(Node1, Ct):-

imp(s, Use, Node1, Sign, Node2, _, _, C1),

allinfer(Node2, C2),

find_multiplier(Sign, Mult, dummy, 0), CS = Mult * C2,

qualifier(Use, CS, Qmult), Ct = CS * C1 * Qmult.

/* An implication with an AND in the Premise */

infer(Node1, Ct):-

imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1),

allinfer(Node2, C2),

allinfer(Node3, C3),

find_multiplier(SignL, MultL, SignR, MultR),

C2S = MultL * C2, C3S = MultR * C3,

min(C2S, C3S, CX), qualifier(Use, CX, Qmult), Ct = CX * C1 * Qmult.

/* An implication with an OR in the Premise */

infer(Node1, Ct):-

```
    imp(o, Use, Node1, SignL, Node2, SignR, Node3, C1),
    allinfer(Node2, C2),
    allinfer(Node3, C3),
    find_multiplier(SignL, MultL, SignR, MultR),
    C2S = MultL * C2, C3S = MultR * C3,
    max(C2S, C3S, CX), qualifier(Use, CX, Qmult), Ct = CX * C1 * Qmult.
```

infer(Node1, Ct):-

```
    terminal_node(Node1), evidence(Node1, Ct),!.
```

infer(Node1, Ct):-

```
    terminal_node(Node1),
    write("What is the certainty for node", Node1),
    nl, readreal(Ct), asserta(evidence(Node1, Ct)),!.
```

/* This is used for simple implication */

```
find_multiplier(pos, 1, dummy, 0).
```

```
find_multiplier(neg, -1, dummy, 0).
```

/* This is used for AND and OR implications */

```
find_multiplier(pos, 1, pos, 1).
```

```
find_multiplier(pos, 1, neg, -1).
```

```
find_multiplier(neg, -1, pos, 1).
```

```
find_multiplier(neg, -1, neg, -1).
```

supercombine([Ct], Ct):-!.

supercombine([C1, C2], Ct):- combine([C1, C2], Ct), !.

supercombine([C1, C2 | T], Ct):- combine([C1, C2], C3), append([C3], T, TL),
supercombine(TL, Ct), !.

combine([-1, 1], 0).

combine([1, -1], 0).

Combine([C1, C2], Ct):- C1 >= 0, C2 >= 0, Ct = C1 + C2 - C1 * C2.

Combine([C1, C2], Ct):- C1 < 0, C2 < 0, Ct = C1 + C2 + C1 * C2.

combine([C1, C2], Ct):- C1 < 0, C2 >= 0, absvalue(C1, Z1), absvalue(C2, Z2),
min(Z1, Z2, Z3), Ct = (C1 + C2) / (1 - Z3).

combine([C1, C2], Ct):- C2 < 0, C1 >= 0, absvalue(C1, Z1), absvalue(C2, Z2),
min(Z1, Z2, Z3), Ct = (C1 + C2) / (1 - Z3).

absvalue(X, Y):- X = 0, Y = 0, !.

absvalue(X, Y):- X > 0, Y = X, !.

absvalue(X, Y):- X < 0, Y = -X, !.

qualifier(Use, C, Qmult):- Use = "r", Qmult = 1, !.

qualifier(Use, C, Qmult):- Use = "n", C >= 0, Qmult = 1, !.

qualifier(Use, C, Qmult):- Use = "n", C < 0, Qmult = 0, !.

Systems that Explain their Actions

/* For and implication, the other in the same manner */

infer(Node1, Ct):-

```
    imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1),
    asserta(dbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)),
    asserta(tdbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)),
    allinfer(Node2, C2),
    allinfer(Node3, C3),
    find_multiplier(SignL, MultL, SignR, MultR),
    C2S = MultL * C2, C3S = MultR * C3,
    min(C2S, C3S, CX), qualifier(Use, CX, Qmult), Ct = CX * C1 * Qmult,
    assertz(infer_summary(
    imp(a, Use, Node1, SignL, Node2, SignR, Node3, C1), Ct)),
    retract(dbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)),
    retract(tdbimp(a, Use, Node1, SignL, Node2, SignR, Node3, C1)).
```

/* How Facility Sub Program */

Exsys_driver :- getallans, showresults, !.

Getallans :- not(prepare_answer).

Prepare_answer :- answer(X, Y), fail.

answer(X, Y) :- hypothesis_node(X), allinfer(X, Y), assert(danswer(X, Y)).

Showresults :- not(displayall).

displayall :- display_aoe_answer, fail.

display_aoe_answer :- danswer(X, Y), clearwindow,
write("For this hypothesis:"), nl,
write(" ", X),nl, write("The certainty is:", Y),nl, nl,
not(how_describer(X)).

how_describer(Node) :- repeat, nl,

write("Type h(how) nodename, or c(to continue),"),
nl, readln(Reply), nl, how_explain(Reply),!.

how_explain(Reply) :- Reply = "c".

how_explain(Reply) :- fronttoken(Reply, _, X1), fronttoken(X1, X, _),
infer_summary(imp(_ _ X, _ _ _ _ _), _), clearwindow,!,
write("The rule(s) that bear upon this conclusion are: "),
nl, nl, infer_summary(imp(A, A1, X, R, S, C, D, E),F),
write("Concluded: ", X), nl, gettype(A, Z),
write("from an ", Z), nl, write(" premise 1 was: ",S), nl,
write(" premise 2 was: ",D), nl,
write("The certainty from use of this rule alone was: ",F),
nl, nl, fail.

```

how_explain(Reply) :- fronttoken(Reply, _, X1), fronttoken(X1, X, _),
    terminal_node(X), evidence(X, C),
    write("You told me that: "), nl, write(" ", X), nl,
    write("has a certainty of: ", C), nl, fail.

```

/* Why Facility Sub Program */

```

infer(Node, Ct) :- terminal_node(Node), evidence(Node, Ct), !.

```

```

infer(Node, Ct) :- terminal_node(Node), repeat, nl,
    write("Type w(why) or give the certainty for node ", Node),
    nl, readln(Reply), reply_to_input(Node, Reply, Ct), !.

```

```

reply_to_input(Node, Reply, Ct) :- not(isname(Reply)), adjuststack,
    str_real(Reply, CT), asserta(evidence(Node, Ct)), !.

```

```

reply_to_input(_, Reply, _) :- isname(Reply), Reply = "w", nl,
    dbimp(U, V, R, S, S1, X, Y, Y1),
    why_describer(U, V, R, S, S1, X, Y, Y1),
    retract(dbimp(U, V, R, S, S1, X, Y, Y1)),
    putadjustflag, pauser, !, fail.

```

```

why_describer(U, U1, V, R, S, X, Y, Z) :- clearwindow, nl, U <>"s", gettype(U, UU),
    write("I am trying to use an inference rule of the type "),
    nl, write(UU), write(", to support the conclusion: "), nl,
    write(" ", V), nl, write("Premise 1 is: ", S), nl, getmode(R, RR),

```

```
write(" This premise will be used ", RR), nl, write("Premise 2 is: ",Y),
nl, getmode(X, XX), nl, write(" This premise will be used ", XX), nl,
write("The certainty of the implication is: ", Z), nl, !.
```

```
why_describer("s", V1, V, R, S, X, Y, Z) :- clearwindow, nl,
write("I am trying to use an inference rule of the type "), nl,
write("simple implication, to support the conclusion: "), nl,
write(" ", V), nl, write("premise 1 is: ", S), nl, getmode(R, RR),
write(" This premise will be used ", RR), nl
write("The certainty of the implication is: ", Z), nl, !.
```

```
gettype("a", "and implication").
```

```
gettype("o", "or implication").
```

```
gettype("s", "simple implication").
```

```
Getmode("pos", "as you see it.").
```

```
Getmode("neg", "prefaced by not.").
```


Natural Language Interfaces
Formal Method

The people respect clever student.

Clever students can own respecting by their good works.

- 1- Build the Context Free Grammar for the above sentences.
- 2- Write a complete prolog program that parse the above sentences using the Context Free Grammar in step 1 .

1.

S → Np, Vp, Np / Np, Vp, Np, Pp

Np → det, noun / adj, noun / noun / det, adj, noun

Vp → verb / h.verb, verb

Pp → preposition, Np

2.

clauses

run:- readln(S), str_to_list(S, L), parse(L).

parse(L):- append(A1, A2, A3, _, L),

 np(A1),

 vp(A2),

 np(A3).

parse(L):- append(A1, A2, A3, A4, L),

 np(A1),

 vp(A2),

 np(A3),

 pp(A4).

np(X):- append(Y1, Y2, _, _, X),

 det(Y1),

 noun(Y2).

np(X):- append(Y1, Y2, _, _, X),

 adj(Y1),

```
        noun(Y2).
np(X):- append(Y1, Y2, Y3, _, X),
        det(Y1),
        adj(Y2),
        noun(Y3).
np(X):- noun(X).

vp(Z):- append(Y1, Y2, _, _, Z),
        h.verb(Y1),
        verb(Y2).
vp(Z):- verb(Z).

pp(M):- append(W1, W2, _, _, M),
        preposition(W1),
        np(W2).
```

```
/* set of Facts */
det(["the"]).  det(["their"]).
noun(["people"]).  noun(["student"]).
noun(["respecting"]).  noun(["work"]).
adj(["clever"]).  adj(["good"]).
verb(["respect"]).  verb(["own"]).
h.verb(["can"]).
preposition(["by"]).
```

Analyzing the semantic structure of a Sentence

Introduction to Thematic Analysis (Case Grammar)

- Object Case (is the noun group that receives the action of the verb)
- Agent Case (is the entity that applies the action to the object)
- Co Agent Case (shares in applying the action that the sentence is about) or (pronoun followed by a noun)

EX: “**The Realtor and his assistant inspected a house for their client .**”

- Beneficiary Case (concerns the entity on whose behalf the action in the sentence was Performed) the beneficiary noun group is “for their client”
- Location Case (concerns noun group that express where the action took place)
- Time Case (this noun group expresses when the action took place)

EX: “**at 5 o’clock**”

- Instrument Case (noun group that identifies something used by the agent to apply the action carried by the verb)

EX: “**with the sharp Knife**”

- Source and Destination Case (the action sentence frequently is about movement from one place or state to another, these beginning and ending places for the action are associated with source and destination noun groups)

EX: “**The dog chased the insurance agent out of the yard and into his car**”

The source case noun group is “**out of the yard**”

The destination group is “**into his car**”

- Trajectory Case (there will be noun groups whose function in the sentence is to describe the path over which the action occurred)

EX: “**The man drove in his car through the woods to his next client**”

- **Conveyance Case** (if the action occurs in some kind of Carrier, this is a conveyance noun group)

EX: “in his car”

Automatic Translation

An Example of the Use of Thematic Analysis (From English Language)

EX: “Jane repaired the radio for Dan with the test instrument“

Verb: (to repair)

Verb tense: past tense

Verb Aspect: 3rd person singular (repaired)

Object: the radio

Agent: Jane

Instrument: the test instrument

Beneficiary: Dan

To Germany Language

Verb: (reparieren)

Verb tense: past tense

Verb Aspect: 3rd person singular (hat repariert)

Object: das radio

Agent: Jana

Instrument: die Probeinstrumenten

Beneficiary: Dan

<agent> <verb_first_part> fur <beneficiary> <object> mit <instrument>
<verb_second_part>

<agent> <verb_first_part> fur <beneficiary> <object>

Jana hat fur Dan das Radio

mit <instrument> <verb_second_part> mit die
Probeinstrumenten repariert

Parts of the Program (Thematic Analysis)

sentence(S,S0) :- agent(S,S1), backparta(S1,S0).

backparta(S,S0) :- verb(S,S1), object(S1, S0).

sentence(S,S0) :- agent(S,S1), backpartb(S1,S0).

backpartb(S,S0) :- verb(S,S1), backpartc(S1, S0).

backpartc(S,S0) :- object(S, S1), instrument(S1,S0).

sentence(S,S0) :- agent(S,S1), backpartd(S1,S0).

backpartd(S,S0) :- verb(S,S1), backparte(S1, S0).

backparte(S,S0) :- object(S, S1), backpartf(S1,S0).

backpartf(S,S0) :- trajectory(S,S1), time(S1,S0).

Natural Language Interfaces
Informal Method (Dictionary Building)

clauses

/* set of facts */

Own(John, B.Sc, 1980, Scientific).
Own(Roy, M.Sc, 1984, technique).
Own(Tomy, B.Sc, 1982, Engineer).
Own(Har, Ph.D, 1978, Scientific).

reject("HOW").
reject("GO").
reject("ALL").
reject("FIND").
reject("THE").
reject("SOME").
reject("I").
reject("HAVE").

dsyn("B.Sc", "B. of Science").
dsyn("M.Sc", "Master of Science").
dsyn("Ph.D", "Philosophy of Doctorate").

docdriver:- repeat, nl, getquery(X), findref(X, Y),
 produceans(Y), fail.

getquery(Z):- write("please ask your question."),
 nl, readln(Y), upper_lower(Y1, Y),
 changeform(Y1, Z).

changeform(S, [H|T]):- fronttoken(S, H, S1), !, changeform(S1, T).
changeform(_, []):-!.

findref(X, Y):- memberof(Y, X), not(reject(Y)), !.

```
produceans(X):- own(X, X1, Y, Z), putflag,  
               write(X, "has", X1, "since the year", Y, "in", Z),nl.  
produceans(X):- syn(X1, W), own(X, W, Y, Z), putflag,  
               write(X, "has", X1, "since the year", Y, "in", Z),nl.  
produceans(_):- not(flag),  
               write("we have no information on that."), nl.  
produceans(_):- remflag.
```

```
putflag:- not(flag), assert(flag),!.  
putflag.
```

```
remflag:- flag, retract(flag),!.  
remflag.
```

```
syn(Y,X):- dsyn(X, Y).  
syn(Y,X):- dsyn(Y, X).
```

```
dsyn(Y,X):- concat(X, "S", Y).  
dsyn(Y,X):- concat(X, "ES", Y).  
dsyn(Y,X):- concat(X, "'S", Y).
```

Computer Sciences Department
(Software & Computer Security Branches)
AI Applications & Systems - Fourth Class

What heuristics would you use in solving these problems?

1. You are looking for a parking space in a moderately crowded parking lot.
 2. You think a particular radio show you want to hear is on now, but you do not know where it is on the dial, and you have no other guidance such as a newspaper listing.
 3. You are in a large office building. You are lost, and you want to find the personal office, but you are embarrassed to ask where it is.
-

Think about an elevator with the following controls: buttons for three floors, buttons to open and close the door, a sensor to see if the door is obstructed, a timer to time how long to leave it open, and single call buttons on each floor. Write a production system that would cause the elevator to operate in the conventional manner if the production system were controlling the operation. Atypical production would be:

If (timer_expired and door_is_open and door_not_obstructed) then
(close_door)

Consider the category scheme to classify expert system. For each of the following, discuss if the example would be an expert system at all and, if so, what type:

1. A program to forecast the local weather.
2. A program to reason about what to do when your car will not started.
3. A program for a help-line service where the person answering the phone has to give advice about poisons that someone might have taken.
4. A program to predict what courses to give and how many sections to plan for in the next three semesters in a large college department.

5. A program to determine the best route for a salesperson to take on any given day to visit all his clients and use the minimum amount of gasoline that is possible.
6. A program to produce a 3-dimensional drawing of a house, given a textual description of the arrangement and dimensions of the rooms.

Write a small expert system program to construct optimal restaurant menus that follows the pattern of Student Advisor System.

The chemical synthesis program currently works with reactions like this:

$X + y \rightarrow z$ with cost (c)

1. How would things have to be modified so that reactions like this one could be included in the reaction data base that the program knows about?

$r \rightarrow s$ with cost (c)

This is anticipating the type of reaction where you treat a chemical in a certain way (heating perhaps) and it turns into something else.

2. How would things have to be modified so that reactions like this one could be included? $q + r + s \rightarrow w$ with cost (c)
3. What modification would be necessary for the program to carry along two costs with each synthesis: One might be the reaction cost and the other the length of time the reaction took to complete.
4. What modification would be necessary for the program to include a function that carries the best synthesis among many syntheses?

Computer Sciences Department
(Software & Computer Security Branches)
AI Applications & Systems - Fourth Class

1. Try to build again the structure of the fuzzy net program (certainty Program) that accept any arbitrary inference tree.
-

2. Given the following information:
index("book1", "OGOFF", ["99"]).
index("book1", "EDLIN", ["41-46", "57"]).
index("book2", "EDLIN", ["100-102"]).
index("book7", "EDLIN", ["100", "110"]).
index("book1", "EXE", ["14-18"]).
index("book1", "ECHO", ["35", "146"]).
index("book7", "BNF", ["51", "55-56"]).
index("book7", "BNF", ["30-31"]).
index("book8", "BNF", ["109", "130-148"]).
index("book4", "RERURSION", ["56-78"]).
index("book7", "RECURSION", ["119-125"]).

dsyn("LOGOUT", "LOGOFF").
dsyn("LOGIN", "LOGON").
dsyn("BENEFIT", "ADVANTAGE").
dsyn("PROCESSING", "MANIPULATING").
dsyn("INTELLIGENT", "SMART").
dsyn("FACT", "REAL").

reject("HOW").
reject("ANY").
reject("ABOUT"). and so on

Write a complete prolog program to index the above information by using the Dictionary (informal) Natural Language Interface technique.

3. Which rules (in the chemical synthesis program) is adjusted when the user asks **how** after the program implementation? Write them.
-

4. Which rules (in the B.B.F program) is adjusted when the user asks **why** when the system asks for any feature? Write them.

University of Technology
Department of Computer Science
Fourth Class (Software & Computer Security Branches)
Lecturer: Dr. Hasanen S. Abdullah
Intelligent Systems & Applications

References

- 1- Fundamentals of Neural Networks: Architecture, Algorithms and Application. By Laurence Fausett.
- 2- Genetic Algorithms (Search, Optimization and Machine Learning). By David E. Goldberg.
- 3- Neural Networks. Fundamentals, Application and Examples. By Werner Kinnebrock.
- 4- Neural Network for Identification, Prediction and Control. By D. T. Pham and X. Liu.

1.1 Introduction

Artificial neural network (ANN) models have been studied for many years with the hope of achieving "Human-like performance", Different names were given to these models such as:

- Parallel distributed processing models
- Biological computers or Electronic Brains.
- Connectionist models
- Neural morphic system

After that, all these names settled on Artificial Neural Networks (ANN) and after it on neural networks (NN) only.

There are two basic different between computer and neural, these are:

- 1- These models are composed of many non-linear computational elements operating in parallel and arranged in patterns reminiscent of biological neural networks.
- 2- Computational Elements (or node s) are connected via weights that are typically adapted during use to improve performance just like human brain.

Computer	→	logic Elements (1, 0)
Neural	→	weighted performance

1.2 Development of Neural Networks

An early attempt to understand biological computations was stimulated by McCulloch & Pitts in [1943], who modeled biological neurons as logical as logical decision elements these elements were described by a two – valued state variables (on, off) and organized into logical decision networks that could compute simple Boolean functions.

In 1961 Rosenblatt solved simple pattern recognition problems using perceptrons. Minsky and Papert in [1969] studied the capabilities and limitations of perceptrons and concluded that many interesting problems could never be solved by perceptron networks.

Recent work by Hopfield examined the computational power of a model system of two –state neurons operating with organized symmetric connections and feed back connectivity. The inclusion of feed –back connectivity in these networks distinguished them from perceptron – line networks. Moreover, graded – response neurons were used to demonstrate the power * speed of these Networks. Recent interest in neural networks is due to the interest in building parallel computers and most importantly due the discovery of powerful network learning algorithms.

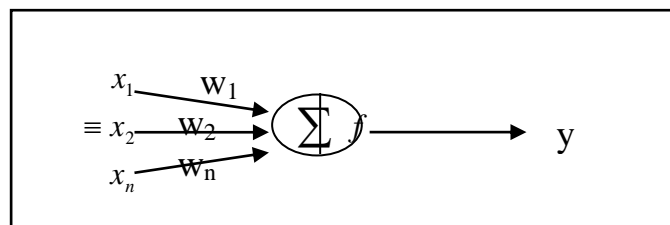
1.3 Areas of Neural Networks

The areas in which neural networks are currently being applied are:

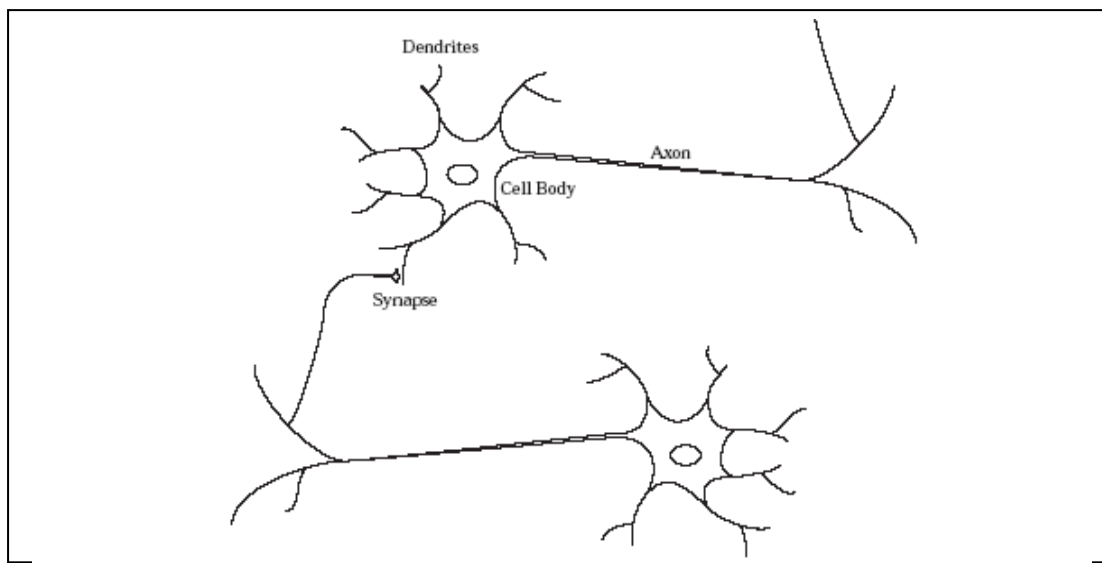
- 1-signal processing
- 2- Pattern Recognition.
- 3-control problems
- 4-medicine
- 5-speech production
- 6-speech Recognition
- 7-Business

2.1 Theory of Neural Networks (NN)

Human brain is the most complicated computing device known to a human being. The capability of thinking, remembering, and problem solving of the brain has inspired many scientists to model its operations. Neural network is an attempt to model the functionality of the brain in a simplified manner. These models attempt to achieve "good" performance via dense interconnections of simple computational elements. The term (ANN) and the connection of its models are typically used to distinguish them from biological network of neurons of living organism which can be represented systematically as shown in figure below



Artificial Neural Network



Biological Neural Network

Neclues is a simple processing unite which receives and combines signals from many other neurons through input paths called **dendrites** if the combined signal is strong enough, it activates the firing of neuron which

produces an o/p signal. The path of the o/p signal is called the **axon**, **synapse** is the junction between the (axon) of the neuron and the dendrites of the other neurons. The transmission across this junction is chemical in nature and the amount of signal transferred depends on the synaptic strength of the junction. This synaptic strength is modified when the brain is learning.

Weights (ANN) \equiv synaptic strength (biological Networks)

2.2 Artificial Neural Networks (ANN)

An artificial neural network is an information processing system that has certain performance characters in common with biological neural networks. Artificial neural networks have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions that:-

- 1-Information processing occurs at many simple elements called neurans.
- 2-Signals are passed between neurons over connection links.
- 3-Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.
- 4-Each neuron applies an action function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

A Neural network is characterized by:

- 1- Architecture: - its pattern of connections between the neurons.
- 2- Training Learning Algorithm: - its method of determining the weights on the connections.
- 3- Activation function.

2.2.1 Properties of ANN

- 1-parallelism
- 2-capacity for adaptation "learning rather programming"
- 3-capacity of generalization
- 4-no problem definition
- 5-abstraction & solving problem with noisy data.
- 6-Ease of construction & learning.
- 7-Distributed memory
- 8- Fault tolerance

2.3 Learning in Neural Network

In case a neural network is to be used for particle applications, a general procedure is to be taken, which in its various steps can be described as follows:-

- 1:** A logical function to be represented is given. The input vector $e_1, e_2, e_3, \dots, e_n$ are present, whom the output vectors $a_1, a_2, a_3, \dots, a_n$ assigned. These functions are to be represented by a network.
- 2:** A topology is to be selected for the network.
- 3:** The weights w_1, w_2, w_3, \dots are to be selected in such away that the network represents The given function (n) the selected topology. Learn procedures are to be used for determining the weights.
- 4:** After the weights have been learned and the network becomes available, it can be used as after as desired.

The learning of weights is generally done as follows:

- 1- Set random numbers. For all weights.
- 2- Select a random input vector e_j .
- 3- Calculate the output vector O_j with the current weights.
- 4- Compare O_j with the destination vector a_j , if $C_j = a_j$ then continue with (2).

Else correct the weights according to a suitable correction formula and then continue with (2).

There are *three type* of learning in which the weights organize themselves according to the task to be learnt, these types are:-

1. Supervised learning

The supervised is that, at every step the system is informed about the exact output vector. The weights are changed according to a formula (e.g. the delta-rule), if o/p is unequal to a. This method can be compared to learning under a teacher, who knows the contents to be learned and regulates them accordingly in the learning procedure.

2. Unsupervised Learning

Here the correct final vector is not specified, but instead the weights are changed through random numbers. With the help of an evaluation function one can ascertain whether the output calculated with the changed weights is better than the previous one. In this case the changed weights are stored, else forgotten. This type of learning is also called reinforcement learning.

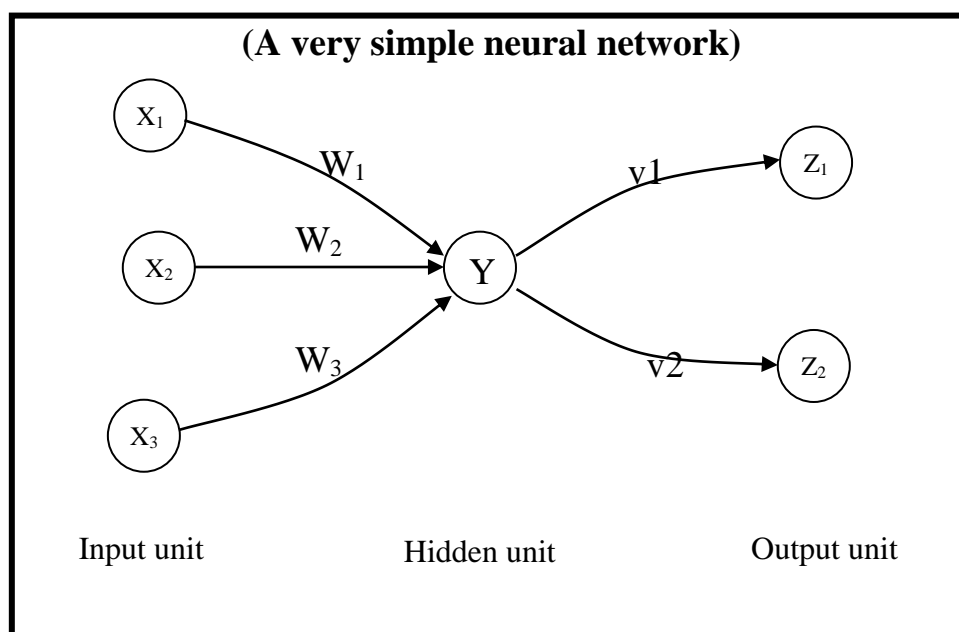
3. Learning through Self- Organization

The weights changed themselves at every learning step. The change depends up on

- 1- The neighborhood of the input pattern.
- 2- The probability pattern, with which the permissible input pattern is offered.

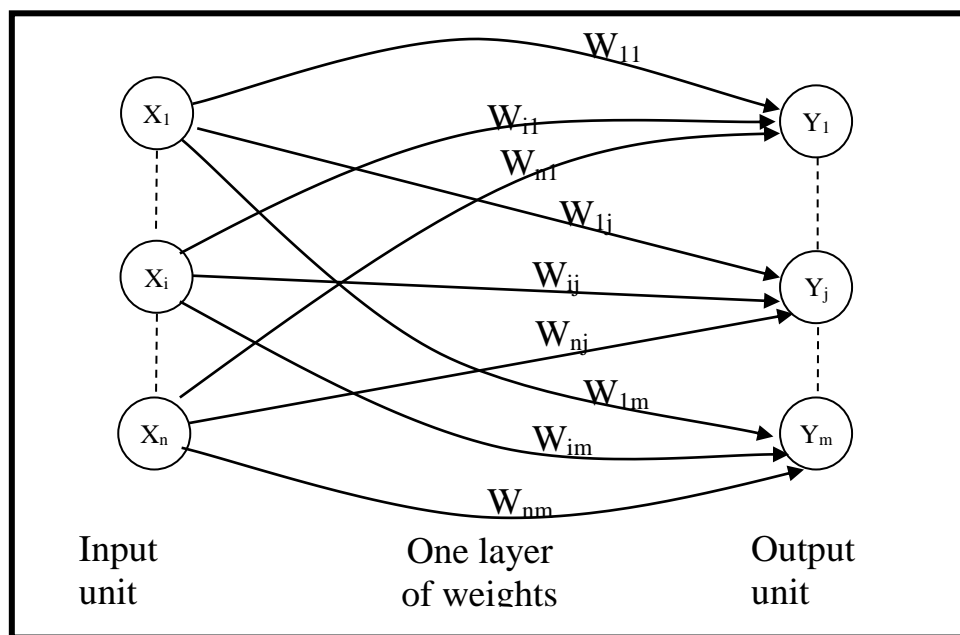
2.4 Typical Architecture of NN

Neural nets are often classified as single layer or multilayer. In determining the number of layers, the input units are not counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnects links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information. The net shown bellow has two layers of weights:



2.4.1 Single-Layer Net:-

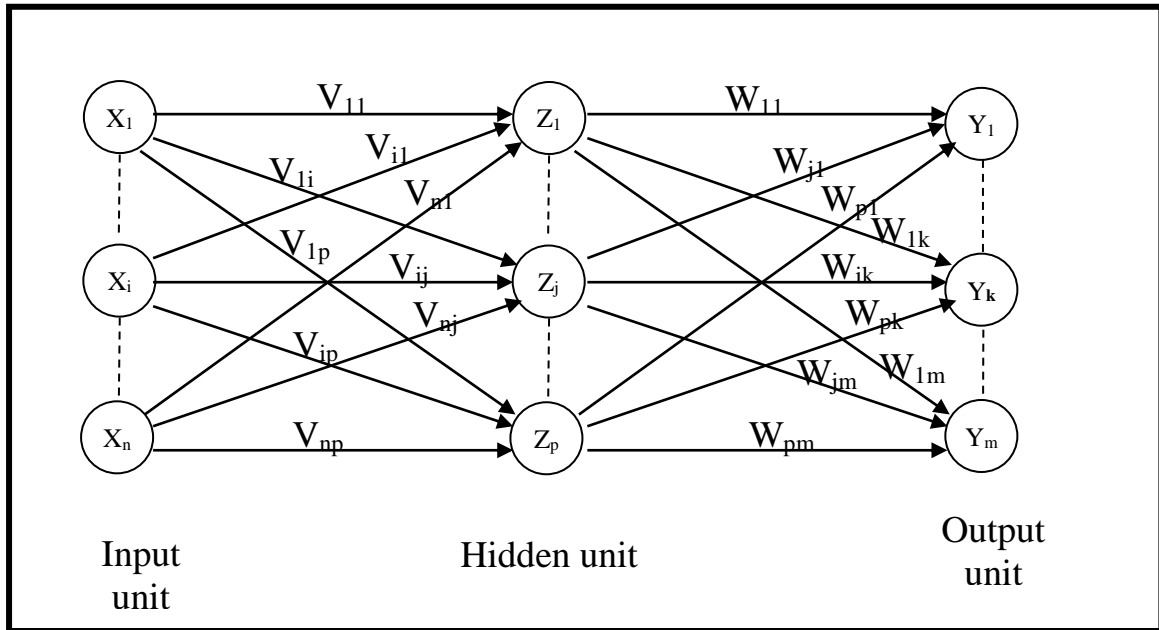
A single-layer net has one layer of connection weight. Often, the units can be distinguished as input units, which receive signals from the outside world, and output units, from which the response of the net can be read. In the typical single-layer net shown in figure bellow the input units are fully connected to output units but are not connected to other input units and the output units are not connected to other output units.



(A single-layer neural network)

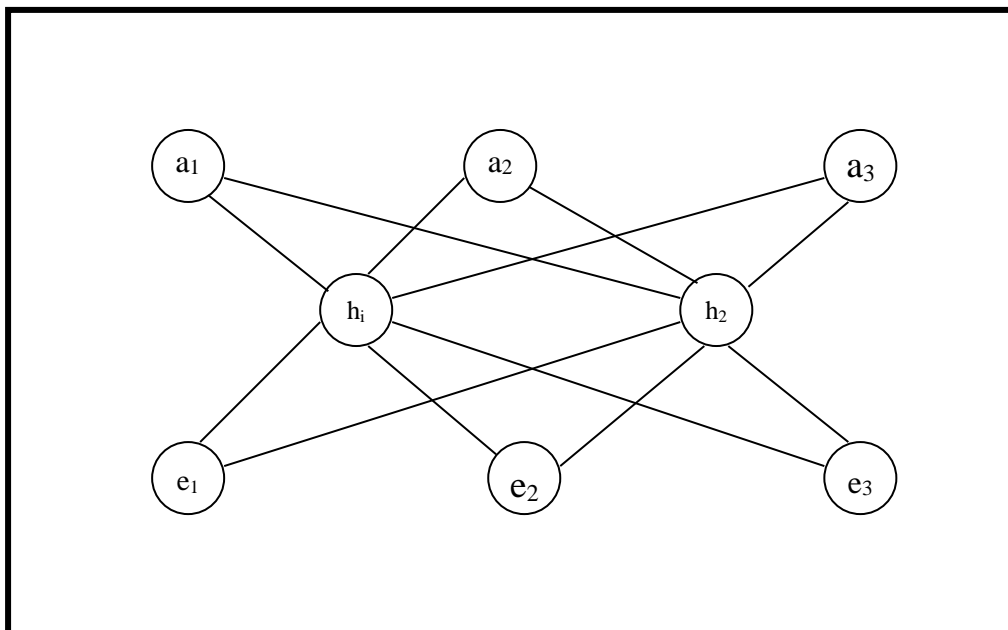
2.4.2 Multilayer net

A Multilayer net is a net with one or more layers (or levels) of nodes which is called hidden units, between the input units and the output units. Typically, there is a layer of weights between two adjacent levels of units (input, hidden, or output). Multilayer nets can solve more complicated problems than can single-layer nets, but training may be more difficult. However, in some cases, training may be more successful because it is possible to solve a problem that a single-layer net can not be trained to perform correctly at all. The figure bellow shows the multilayer neural net.



(A Multilayer Neural Net)

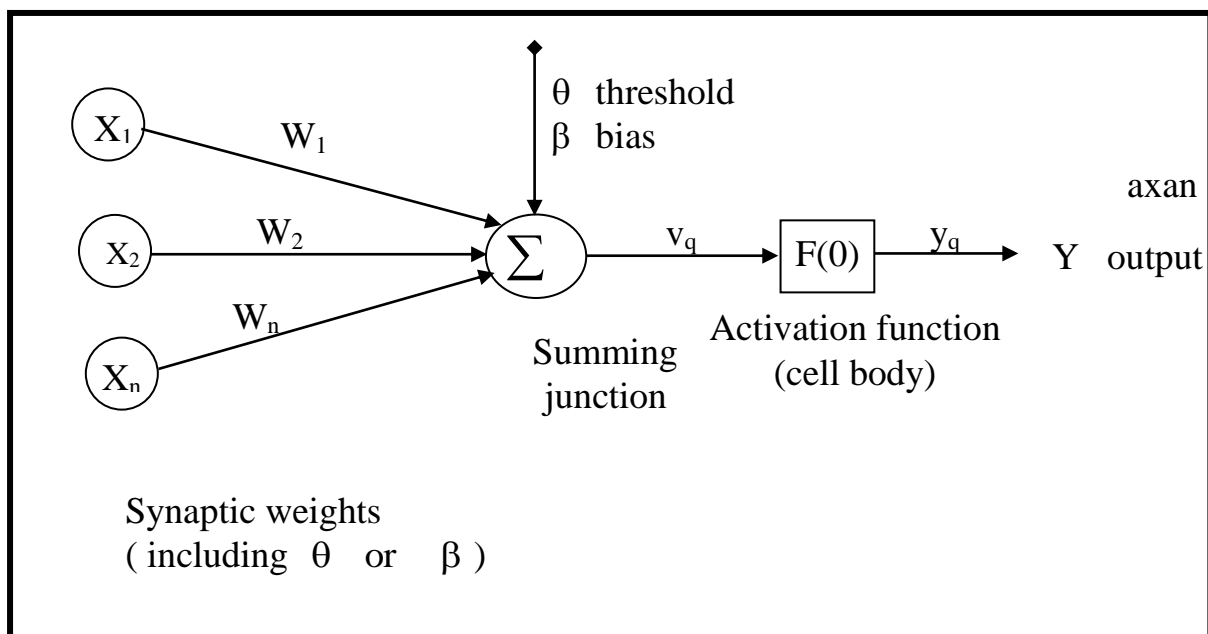
The figure shown below is an example of a three-layered neural net work with two hidden neurons.



2.5 Basic Activation Functions

The activation function (Sometimes called a transfers function) shown in figure below can be a linear or nonlinear function. There are many different types of activation functions. Selection of one type over another depends on the particular problem that the neuron (or neural network) is to solve. The most common types of activation function are:-

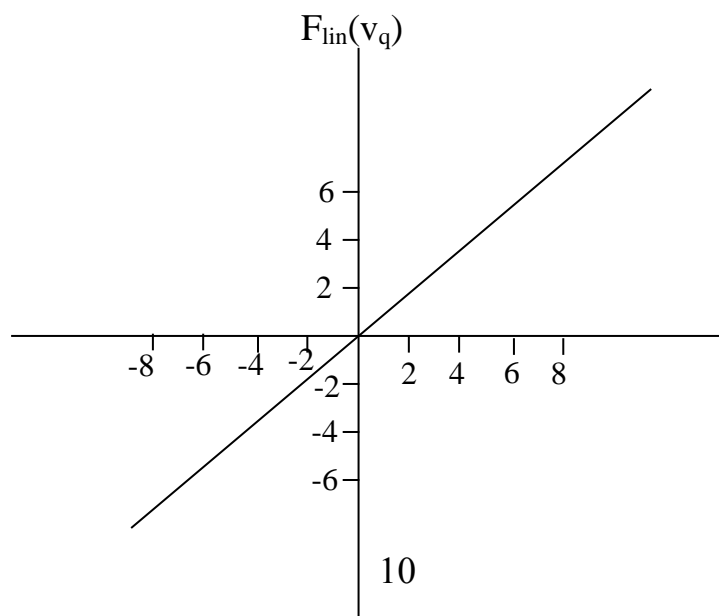
$$V_q = \sum_{j=0}^n W_{qj} X_j$$



Alternate nonlinear model of an ANN

1- The first type is *the linear* (or *identity*) function. Ramp

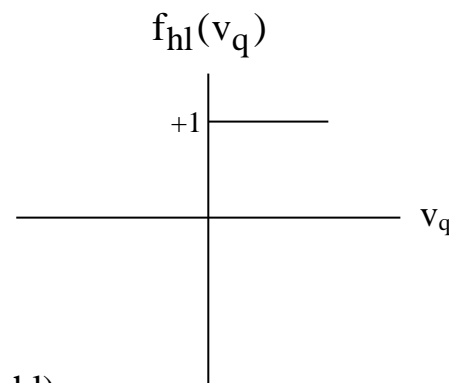
$$y_q f_{lin}(v_q) = v_q$$



2-The second type of activation function is a **hard limiter**; this is a binary (or bipolar) function that hard-limits the input to the function to either a 0 or a 1 for the binary type, and a -1 or 1 for the bipolar type. The binary hard limiter is sometimes called the threshold function, and the bipolar hard limiter is referred to as the symmetric hard limiter.

a- The o/p of the binary hard limiter:-

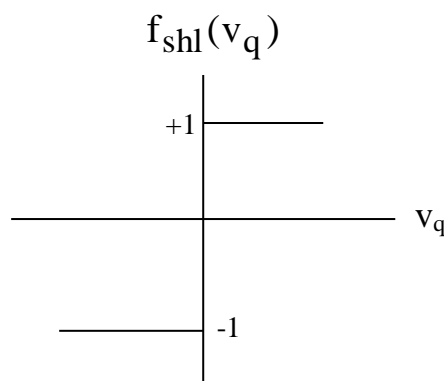
$$y_q = f_{hl}(v_q) = \begin{cases} 0 & \text{if } v_q < 0 \\ 1 & \text{if } v_q \geq 0 \end{cases}$$



b-The o/p for the symmetric hard limiter (shl):-

$$y_q = f_{shl}(v_q) = \begin{cases} -1 & \text{if } v_q < 0 \\ 0 & \text{if } v_q = 0 \\ 1 & \text{if } v_q > 0 \end{cases}$$

تسمى ايضا double side



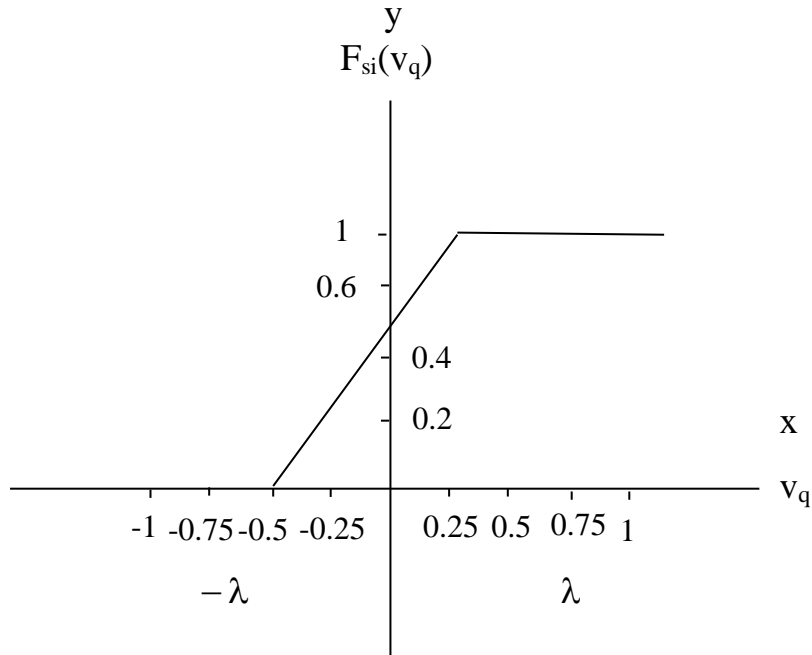
3-The third type of basic activation function is the **saturation linear** function or threshold logic Unite (tLu) .

This type of function can have either a binary or bipolar range for the saturation limits of the output. The bipolar saturating linear function will be referred to as the symmetric saturating linear function.

a- The o/p for the *saturation linear* function (binary o/p):-

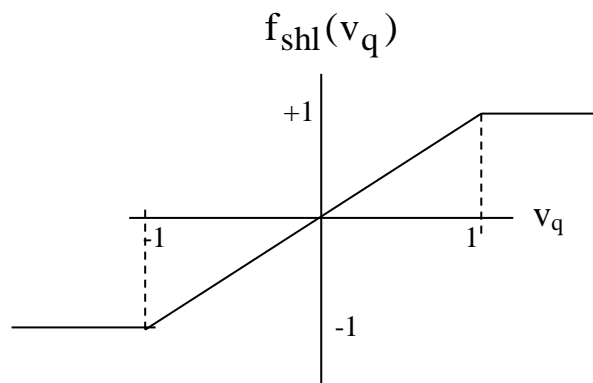
$$y_q = f_{sl}(v_q) = \begin{cases} 0 & \text{if } v_q < -1/2 \\ v_q + 1/2 & \text{if } -1/2 \leq v_q \leq 1/2 \\ 1 & \text{if } v_q > 1/2 \end{cases}$$

or
$$y = \begin{cases} \lambda & \text{if } x > \lambda \\ x & \text{if } -\lambda \leq x \leq \lambda \\ -\lambda & \text{if } x < -\lambda \end{cases}$$



b- The o/p for the *symmetric saturation linear* function:-

$$y_q = f_{ssl}(v_q) = \begin{cases} -1 & \text{if } v_q < -1 \\ v_q & \text{if } -1 \leq v_q \leq 1 \\ 1 & \text{if } v_q > 1 \end{cases}$$

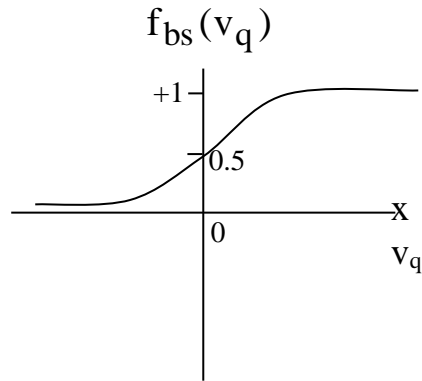


4-The fourth type is **sigmoid**. Modern NN's use the sigmoid nonlinearity which is also known as logistic, semi linear, or squashing function.

محصورة بين 0 و 1 وبها مرونة

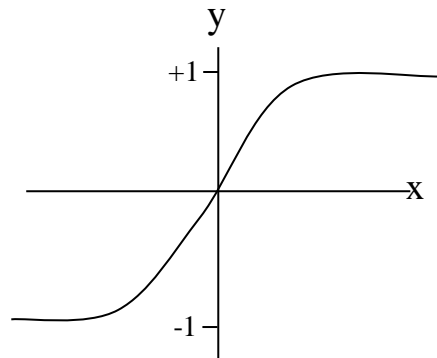
$$y_q = f_{bs}(v_q) = \frac{1}{1 + e^{-\infty v_q}}$$

$$y = \frac{1}{1 + e^{-x}}$$



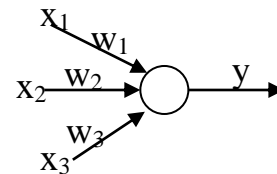
5-Hyperbolic tangent function is similar to sigmoid in shape but symmetric about the origin. (tan h)

$$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Ex.1 find y for the following neuron if :- $x_1=0.5$, $x_2=1$, $x_3=0.7$

$$w_1=0, w_2=-0.3, w_3=0.6$$



Sol

$$\text{net} = X_1 W_1 + X_2 W_2 + X_3 W_3$$

$$= 0.5 * 0 + 1 * -0.3 + (-0.7 * 0.6) = -0.72$$

1- if f is linear

$$y = -0.72$$

2- if f is hard limiter (on-off)

$$y = -1$$

3- if f is sigmoid

$$y = \frac{1}{1 + e^{-(-0.72)}} = 0.32$$

4-if f is tan h

$$y = \frac{e^{-0.72} - e^{0.72}}{e^{-0.72} + e^{0.72}} = -0.6169$$

5-if f is (TLU) with $b=0.6$, $a=3$ then $y=-3$

$$f(y) = \begin{cases} a & y > b \\ ky & -b < y < b \\ -a & y < -b \end{cases} \quad \left| \quad f(y) = \begin{cases} a & y > b \\ ky & 0 < y < b \end{cases}$$

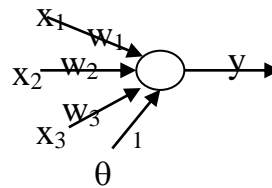
Ex2:- (H.W)

Find y for the following neuron if

$$x_1 = 0.5, \quad x_2 = 1, \quad x_3 = -0.7$$

$$w_1 = 0, \quad w_2 = -0.3, \quad w_3 = 0.6$$

$$\theta = 1$$



Sol

$$\begin{aligned} \text{Net} &= \sum W_i X_i + \theta \\ &= -0.72 + 1 = 0.28 \end{aligned}$$

1- if f is linear

$$y = 0.28$$

2- if f is hard limiter

$$y = 1$$

3-if f is sigmoid

$$y = \frac{1}{1 + e^{-0.28}} = 0.569$$

4-if f is tan sh

$$y = \frac{e^{0.28} - e^{-0.28}}{e^{0.28} + e^{-0.28}} = 0.272$$

5-if f is TLU with $b=0.6, +a=3$

$$y=0.28 \quad y \leftarrow -b < y < b$$

Ex.3

The output of a simulated neural using a sigmoid function is 0.5 find the value of threshold when the input $x_1 = 1, x_2 = 1.5, x_3 = 2.5$. and have initial weights value = 0.2.

Sol

$$\text{Output} = F(\text{net} + \theta)$$

$$F(\text{net}) = \frac{1}{1 + e^{-\text{net}}}$$

$$\text{Net} = \sum W_i X_i$$

$$= X_1 W_1 + X_2 W_2 + X_3 W_3$$

$$= (1 * 0.2) + (1.5 * 0.2) + (2.5 * 0.2) = 0.2 + 0.30 + 0.50 = 1$$

$$0.5 = \frac{1}{1 + e^{-(1+\theta)}}$$

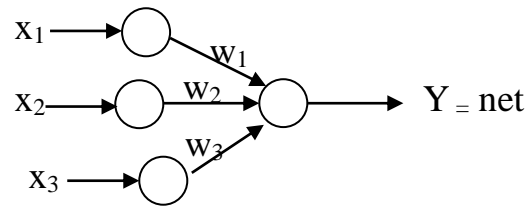
$$0.5 (1 + e^{-(1+\theta)}) = 1$$

$$0.5 + 0.5 e^{-(1+\theta)} = 1$$

$$0.5 e^{-(1+\theta)} = 0.5$$

$$e^{-(1+\theta)} = 1$$

$$-(1+\theta) = \ln 1 \Rightarrow -1 - \theta = 0 \Rightarrow -\theta = 1 \Rightarrow \therefore \theta = -1$$



2.6 The Bias

قيمة ثابتة تضاف لتحسين التعلم

Some networks employ a bias unit as part of every layer except the output layer. This units have a constant activation value of 1 or -1, it's weight might be adjusted during learning. The bias unit provides a constant term in the weighted sum which results in an improvement on the convergence properties of the network.

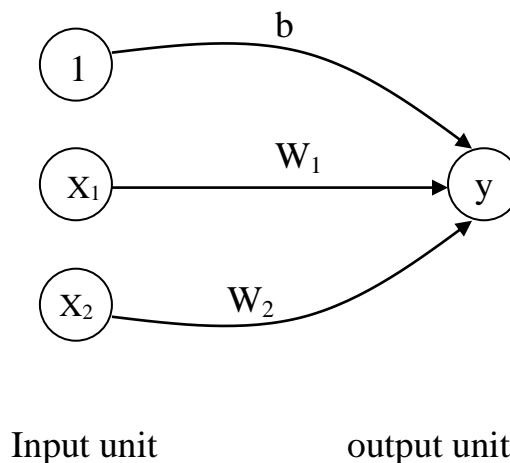
A bias acts exactly as a weight on a connection from a unit whose activation is always 1. Increasing the bias increases the net input to the unit. If a bias is included, the activation function is typically taken to be:

$$f(\text{net}) \begin{cases} 1 & \text{if } \text{net} \geq 0; \\ -1 & \text{if } \text{net} < 0; \end{cases}$$

Where

$$\text{net} = b + \sum_i X_i W_i$$

Figure: - single –layer NN for logic function



Some authors do not use a bias weight, but instead use a fixed threshold θ for the activation function.

$$f(\text{net}) \begin{cases} 1 & \text{if } \text{net} \geq 0; \\ -1 & \text{if } \text{net} < 0; \end{cases}$$

Where

$$\text{net} = b + \sum_i X_i W_i$$

However, this is essentially equivalent to the use of an adjustable bias.

3.1 Learning Algorithms

The NN's mimic the way that a child learns to identify shapes and colors NN algorithms are able to adapt continuously based on current results to improve performance. Adaptation or learning is an essential feature of NN's in order to handle the new "environments" that are continuously encountered. In contrast to NN's algorithms, traditional statistical techniques are not adoption but typically process all training data simultaneously before being used with new data. The performance of learning procedure depends on many factors such as:-

- 1- The choice of error function.
- 2- The net architecture.
- 3- Types of nodes and possible restrictions on the values of the weights.
- 4- An activation function.

The convergent of the net:-

Depends on the:-

- 1- Training set
- 2- The initial conditions
- 3- Learning algorithms.

Note:-

The convergence in the case of complete information is better than in the case of incomplete information

Training a NN is to perform weights assignment in a net to minimize the o/p error. The net is said to be trained when convergence is achieved or in other words the weights stop changing.

The learning rules are considered as various types of the:-

3.1.1 Hebbian Learning Rule

The earliest and simplest learning rule for a neural net is generally known as the Hebb rule. Hebbian learning rule suggested by Hebb in 1949. Hebb's basic idea is that if a unit U_j receives an input from a unit U_i and both unite are highly active, then the weight W_{ij} (from unit i to unit j) should be strengthened.

This idea is formulated as:-

$$\Delta w_{ij} = \zeta x_i y_j$$

Where ζ is the learning rate $\zeta = \alpha = 1$, Δw is the weight change

$$w(\text{new}) = w(\text{old}) + xy$$

$$\therefore w(\text{new}) = w(\text{old}) + \Delta w$$

-: Hebbian learning محاسن -:

تسمح بتقليل قيم الـ weight عندما الـ I/p يكون active والـ o/p active وهذه من السهولة يتم انجازها باستخدام -1 and +1 بدلا من 0 and 1 .

-: Hebbian learning مساوئ الـ -:

Hebbian learning takes no account of the actual value of the output, only the desired value. This limitation can be overcome if the weights are adjusted by amount which depends upon the error between the desired and actual output. This error is called delta, S , and the new learning rule is called the delta rule.

Algorithm (Hebbian learning Rule)**Step 0:** Initialize all weights

$$w_i = 0 \quad (i = 1 \text{ to } n)$$

Step 1: for each I/p training vector target o/p

Pair. S : t do steps 2- 4.

Step 2 : Set activations for I/P units:

$$w_i = s_i \quad (i = 1 \text{ to } n)$$

Step 3 : set activation for O/P unit :

$$y = t$$

Step 4 : Adjust the weights for

$$w_i (\text{new}) = w_i (\text{old}) + x_i y \quad (i = 1 \text{ to } n)$$

Adjust the bias:

$$b(\text{new}) = b(\text{old}) + y$$

Note that the bias is adjusted exactly like a weight from a "unit" whose output signal is always 1.

Ex 4:

A Hebb net for the ABD function: binary input and targets

Input		Target	
1	1	1	1
1	0	1	0
0	1	1	0
0	0	1	0

$$\Delta w_1 = x_1 y, \quad \Delta w_2 = x_2 y, \quad \Delta b = y$$

Initial weights = 0, $w_1 = 0$, $w_2 = 0$, $w_3 = 0$

	x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1 0	w_2 0	b 0
1	1	1	1	1	1	1	1	1	1	1
2	1	0	1	0	0	0	0	1	1	1
3	0	1	1	0	0	0	0	1	1	1
4	0	0	1	0	0	0	0	1	1	1

The first input pattern shows that the response will be correct presenting the second, third, and fourth training i/p shows that because the target value is 0, no learning occurs. Thus, using binary target values prevents the net from learning only pattern for which the target is "off".

The AND function can be solved if we modify its representation to express the inputs as well as the targets in bipolar form. Bipolar representation of the inputs and targets allows modifications of a weight when the input unit and the target value are both "on" at the same time and when they are both "off" at the same time and all units will learn whenever there is an error in the output. The Hebb net for the AND function: bipolar inputs and targets are:

$$\Delta w_1 = x_1 * y$$

$$= 1 * 1 = 1$$

$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1$$

$$= 0 + 1 = 1$$

x₁	x₂	b	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Presenting the first input:-

x₁	x₂	b	y	Δw_1	Δw_2	Δb	w₁	w₂	b
1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1

Presenting the second input:-

x₁	x₂	b	y	Δw_1	Δw_2	Δb	w₁	w₂	b
1	-1	1	-1	-1	1	-1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0

Presenting the third input:-

x₁	x₂	b	y	Δw_1	Δw_2	Δb	w₁	w₂	b
-1	1	1	-1	1	-1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1

Presenting the fourth input:-

x₁	x₂	b	y	Δw_1	Δw_2	Δb	w₁	w₂	b
-1	-1	1	-1	1	1	-1	2	2	-2

The first iteration will be:-

Input			target	Weight change			weights		
x₁	x₂	b	y	Δw_1	Δw_2	Δb	w₁	w₂	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

Second Method

$$W_{ij} = \sum X_i Y_j \quad \text{or} \quad [W] = [X]^T [Y]$$

Ex. 5

What would the weights be if Hebbian learning is applied to the data shown in the following table? Assume that the weights are all zero at the start.

p	x₁	x₂	y
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

With weights that you've just found, what output values are produce with a threshold of 1, using hyperbolic activation function.

	x_1	x_2	y	Δw_1	Δw_2	w_1	w_2
1	0	0	1	0	0	0	0
2	0	1	1	0	1	0	1
3	1	0	0	0	0	0	1
4	1	1	1	1	1	1	2

$$p=0, \quad w_1 = 0, \quad w_2 = 0$$

$$p=1, \quad w_1 = 0 + x_1 * y = 0$$

$$w_2 = 0 + x_2 * y = 0$$

$$p=2, \quad w_1 = 0 + 0 * 1 = 0$$

$$w_2 = 0 + 1 * 1 = 1$$

$$p=3, \quad w_1 = 0 + 1 * 0 = 0$$

$$w_2 = 1 + 0 * 0 = 1$$

$$p=4, \quad w_1 = 0 + 1 * 1 = 1$$

$$w_2 = 1 + 1 * 1 = 2$$

$$\therefore w_1 = 1, \quad w_2 = 2$$

$$\text{net} = \sum_{i=1}^4 X_i * W_i$$

$$p=1, \quad \text{net} = x_1 * w_1 + x_2 * w_2$$

$$= 0 * 1 + 0 * 2 = 0$$

$$p=2, \quad \text{net} = 0 * 1 + 1 * 2 = 2$$

$$p=3, \quad \text{net} = 1 * 1 + 0 * 2 = 1$$

$$p=4, \quad \text{net} = 1 * 1 + 1 * 2 = 3$$

$$\text{output} = F(\text{net} + \theta) = \frac{e^{(\text{net} + \theta)} - e^{-(\text{net} + \theta)}}{e^{(\text{net} + \theta)} + e^{-(\text{net} + \theta)}}$$

p	x₁	x₂	net	y
1	0	0	0	0
2	0	1	2	1
3	1	0	1	0
4	1	1	3	1

3.1.2 Basic Delta Rule (BDR)

The idea of Hebb was modified to produce the widrow-Hoff (delta) rule in 1960 or least Mean Square (LMS). The BDR is formulated as:-

$$\Delta w_{ij} = \xi (d_j - y_i) x_i$$

$$\Delta w_{ij} = \xi \delta_j x_i \quad (\text{Delta rule})$$

Δw : - is the weight change

ξ : - is the learning rate

d :- desired output

y :- actual output

δ : - error between d and y

Note:-

Before training the net, a decision has to be made on the setting of the learning rate. Theoretically, the larger ξ the faster training process goes. But practically, ξ may have to be set to a small value (e.g 0.1) in order to prevent the training process from being trapped at local minimum resulting at oscillatory behavior.

H.W**Q1**:- Briefly discuss the following:

A-Dendrites

B-synapses

Q2:- A fully connected feed forward network has 10 source nodes, 2 hidden layers, one with 4 neurons and other with 3 neurons, and single output neuron. Construct an architecture graph of this network.

Q3:- A neuron j receives input from four other neurons whose activity levels are 10, -20, 4 and -2. The respective synaptic weights of neuron j are 0.8, 0.2, -1, and -0.9. Calculate the output of neuron j for the following two activation functions:-

- i) Hard-limiting function
- ii) Logistic function $F(x) = 1/(1 + e^{-x})$.

Q4:- perform 2 training steps of the Delta learning rules using $\xi = 1$ & the following data specifying the initial weights W_1 , & the two training pairs

$$W_1 = \left\{ \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right\}, \left\{ x_1 = \begin{matrix} 2 \\ 1 \\ -1 \end{matrix} \right\}, d_1 = -1, x_2 = \left\{ \begin{matrix} 0 \\ -1 \\ -1 \end{matrix} \right\}, d_2 = 1 \right\}$$

Q5:- list the features that distinguish the delta rule & Hebb's rule from each other?

3.1.3 Back Propagation

The determination of the error is a recursive process which start with the o/p units and the error is back propagated to the I/p units. Therefore the rule is called error Back propagation (EBP) or simply Back Propagation (BP). The weight is changed exactly in the same form of the standard DR

$$\Delta w_{ij} = \xi \delta_j x_i$$

$$\Rightarrow w_{ij}(t+1) = w_{ij}(t) + \xi \delta_j x_i$$

There are two other equations that specify the error signal. If a unite is an o/p unit, the error signal is given by:-

$$\delta = (d_j - y_j) f_j(\text{net } j)$$

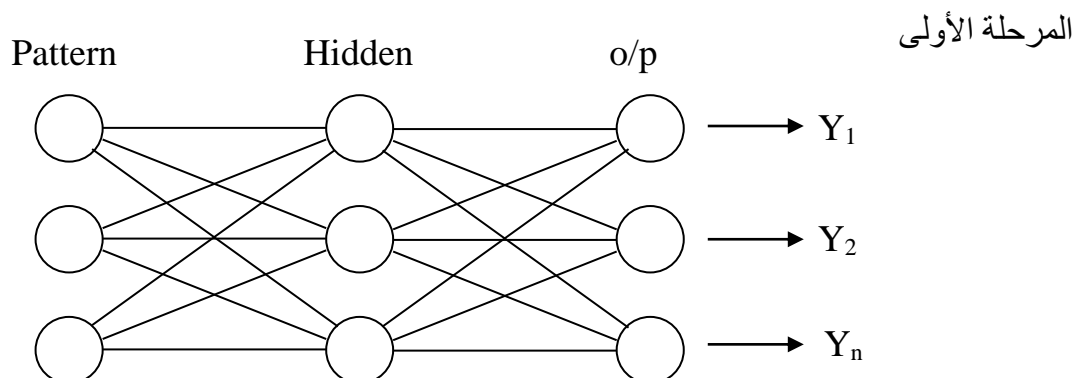
$$\text{Where } \text{net } j = \sum w_{ij} x_i + \theta$$

The GDR minimize the squares of the differences between the actual and the desired o/p values summed over the o/p unit and all pairs of I/p and o/p vectors. The rule minimize the overall error $E = \sum E_p$ by implementing a gradient descent in E: - where, $E_p = 1/2 \sum_j (d_j - y_j)^2$.

The BP consists of two phases:-

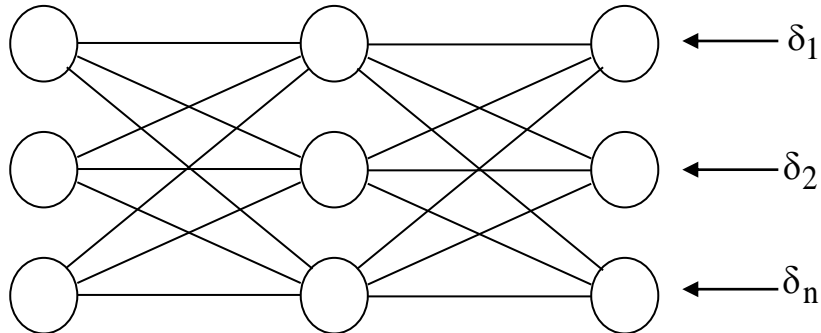
1- Forward Propagation:-

During the forward phase, the I/p is presented and propagated towards the o/p.



2- Backward Propagation:-

During the backward phase, the errors are formed at the o/p and propagated towards the I/p



3- Compute the error in the hidden layer.

$$\text{If } y = f(x) = \frac{1}{1 + e^{-x}}$$

$$f' = y(1 - y)$$

Equation is can rewrite as:-

$$\delta_j = y(1 - y)(d_j - y_j)$$

The error signal for hidden units for which there is no specified target (desired o/p) is determined recursively in terms of the error signals of the units to which it directly connects and the weights of those connections:-

That is

$$\delta_j = f'(\text{net}_j) \sum_k \delta_k w_{jk}$$

Or

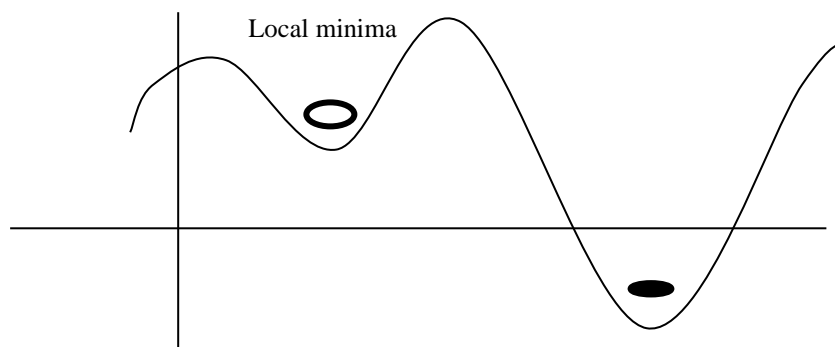
$$\delta_j = y_j(1 - y_j) \sum_k \delta_k w_{jk}$$

B.P learning is implemented when hidden units are embedded between input and output units.

Convergence

A quantitative measure of the learning is the :Root Mean Square (RMS) error which is calculated to reflect the "degree" of learning.

Generally, an RMS below (0.1) indicates that the net has learned its training set. Note that the net does not provide a yes /no response that is "correct" or "incorrect" since the net get closer to the target value incrementally with each step. It is possible to define a cut off point when the nets o/p is said to match the target values.



- Convergence is not always easy to achieve because sometimes the net gets stuck in a "Local minima" and stops learning algorithm.
- Convergence can be represented intuitively in terms of walking about mountains.

Momentum term

The choice of the learning rate plays important role in the stability of the process. It is possible to choose a learning rate as large as possible without leading to oscillations. This offers the most rapid learning. One way to increase the learning rate without leading to oscillations is to modify the GDR to include momentum term.

This can be achieved by the following rule:-

$$W_{ij}(t+1) = W_{ij}(t) + \zeta \delta_j x_i + \alpha (W_{ij}(t) - W_{ij}(t-1))$$

Where α ($0 < \alpha < 1$) is a constant which determines the effect of the past weight changes on the current direction of movement in weight space.

A "global minima" unfortunately it is possible to encounter a local minima, avally that is not the lowest possible in the entire terrain. The net does not leave a local minima by the standard BP algorithm and special techniques should be used to get out of a local minima such as:-

- 1- Change the learning rate or the momentum term.
- 2- Change the no. of hidden units (10%).
- 3- Add small random value to the weights.
- 4- Start the learning again with different initial weights.

3.1.3.1 Back Propagation Training Algorithm

Training a network by back propagation involves three stages:-

1-the feed forward of the input training pattern

2-the back propagation of the associated error

3-the adjustment of the weights

let n = number of input units in input layer,

let p = number of hidden units in hidden layer

let m = number of output units in output layer

let V_{ij} be the weights between i/p layer and the hidden layer,

let W_{ij} be the weights between hidden layer and the output layer,

we refer to the i/p units as X_i , $i=1, 2, \dots, n$. and we refer to the hidden units as

Z_j , $j=1, \dots, p$. and we refer to the o/p units as y_k , $k=1, \dots, m$.

δ_{1j} is the error in hidden layer,

δ_{2k} is the error in output layer,

ζ is the learning rate

α is the momentum coefficient (learning coefficient, $0.0 < \alpha < 1.0$),
 y_k is the o/p of the net (o/p layer),
 Z_j is the o/p of the hidden layer,
 X_i is the o/p of the i/p layer.
 η is the learning coefficient.

The algorithm is as following :-

Step 0 : initialize weights (set to small random value).

Step 1 : while stopping condition is false do steps 2-9

Step 2: for each training pair, do steps 3-8

Feed forward :-

Step 3:- Each i/p unit (X_i) receives i/p signal X_i & broad casts this signal to all units in the layer above (the hidden layer)

Step 4:- Each hidden unit (Z_j) sums its weighted i/p signals,

$$Z - inj = V_{aj} + \sum_{i=1}^n x_i v_{ij} \quad (V_{aj} \text{ is abias})$$

and applies its activation function to compute its output signal (the activation function is the binary sigmoid function),

$$Z_j f(Z - inj) = 1 / (1 + \exp - (Z - inj))$$

and sends this signal to all units in the layer above (the o/p layer).

Step 5:- Each output unit (Y_k)sums its weighted i/p signals,

$$y - ink = w_{ok} + \sum_{j=1}^p Z_j w_{jk} \quad (\text{where } w_{ok} \text{ is abias})$$

and applies its activation function to compute its output signal.

$$y_k = f(y - ink) = 1 / (1 + \exp - (y - ink))$$

back propagation of error:-

step 6 : Each output unit (y_k , $k= 1$ to m) receive a target pattern corresponding to the input training pattern, computes its error information term and calculates its weights correction term used to update W_{jk} later,

$$\delta_{2k} = y_k(1 - y_k) * (T_k - y_k),$$

where T_k is the target pattern & $k=1$ to m .

step 7 : Each hidden unit (Z_j , $j= 1$ to p) computes its error information term and calculates its weight correction term used to update V_{ij} later,

$$\delta_{1j} = Z_j * (1 - Z_j) * \sum_{k=1}^m \delta_{2k} W_{jk}$$

Update weights and bias :-

step 8: Each output unit (y_k , $k=1$ to m) updates its bias and weights:

$$W_{jk}(\text{new}) = \eta * \delta_{2k} * Z_j + \alpha * [W_{jk}(\text{dd})],$$

$j= 1$ to p

Each hidden unit (Z_j , $j= 1$ to p) update its bias and weights:

$$V_{ij}(\text{new}) = \eta * \delta_{1j} * X_i + \alpha [v_{ij}(\text{dd})],$$

$I = 1$ to n

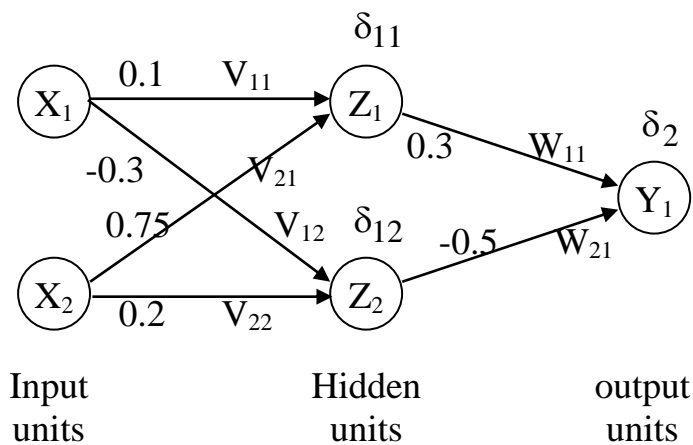
Step 9 : Test stopping condition.

EX6

Suppose you have BP- ANN with 2-input , 2-hidden , 1-output nodes with sigmoid function and the following matrices weight, trace with 1-iteration.

$$V = \begin{bmatrix} 0.1 & -0.3 \\ 0.75 & 0.2 \end{bmatrix} \quad w = [0.3 \quad -0.5]$$

Where $\alpha = 0.9$, $\eta = 0.45$, $x = (1,0)$, and $T_k = 1$

Solution:-**1-Forward phase :-**

$$Z - \text{in}1 = X_1 V_{11} + X_2 V_{21} = 1 * 0.1 + 0 * 0.75 = 0.1$$

$$Z - \text{in}2 = X_1 V_{12} + X_2 V_{22} = 1 * -0.3 + 0 * 0.2 = -0.3$$

$$Z_1 = f(Z - \text{in}1) = 1 / (1 + \exp - (Z - \text{in}1)) = 0.5$$

$$Z_2 = f(Z - \text{in}2) = 1 / (1 + \exp - (Z - \text{in}2)) = 0.426$$

$$y - \text{in}1 = Z_1 W_{11} + Z_2 W_{21}$$

$$= 0.5 * 0.3 + 0.426 * (-0.5) = -0.063$$

$$y_1 = f(y - \text{in}1) = 1 / (1 + \exp - (y - \text{in}1)) = 0.484$$

2-Backward phase :-

$$\delta_{2k} = y_k(1 - y_k) * (T_k - y_k)$$

$$\delta_{21} = 0.484(1 - 0.484) * (1 - 0.484) = 0.129$$

$$\delta_{1j} = Z_j * (1 - Z_j) * \sum_{k=1}^m \delta_{2k} W_{jk}$$

$$\begin{aligned} \delta_{11} &= Z_1(1 - Z_1) * (\delta_{21} W_{11}) \\ &= 0.5(1 - 0.5) * (0.129 * 0.3) = 0.0097 \end{aligned}$$

$$\begin{aligned} \delta_{12} &= Z_2(1 - Z_2) * (\delta_{21} W_{21}) \\ &= 0.426(1 - 0.426) * (0.129 * (-0.5)) = -0.015 \end{aligned}$$

3-Update weights:-

$$W_{jk}(\text{new}) = \eta * \delta_{2k} * Z_j + \alpha * [W_{jk}(\text{old})]$$

$$\begin{aligned} W_{11} &= \eta * \delta_{21} * Z_1 + \alpha * [W_{11}(\text{old})] \\ &= 0.45 * 0.129 * 0.5 + 0.9 * 0.3 = 0.299 \end{aligned}$$

$$\begin{aligned} W_{21} &= \eta * \delta_{21} * Z_2 + \alpha * [W_{21}(\text{old})] \\ &= 0.45 * 0.129 * 0.426 + 0.9 * -0.5 = -0.4253 \end{aligned}$$

$$V_{ij}(\text{new}) = \eta * \delta_{1j} * X_i + \alpha * [V_{ij}(\text{old})]$$

$$\begin{aligned} V_{11} &= \eta * \delta_{11} * X_1 + \alpha * [V_{11}(\text{old})] \\ &= 0.45 * 0.0097 * 1 + 0.9 * 0.1 = 0.0944 \end{aligned}$$

$$\begin{aligned} V_{12} &= \eta * \delta_{12} * X_1 + \alpha * [V_{12}(\text{old})] \\ &= 0.45 * 0.0158 * 1 + 0.9 * -0.3 = -0.2771 \end{aligned}$$

$$\begin{aligned} V_{21} &= \eta * \delta_{11} * X_2 + \alpha * [V_{21}(\text{old})] \\ &= 0.45 * 0.0097 * 0 + 0.9 * 0.75 = 0.675 \end{aligned}$$

$$\begin{aligned} V_{22} &= \eta * \delta_{12} * X_2 + \alpha * [V_{22}(\text{old})] \\ &= 0.45 * -0.0158 * 0 + 0.9 * 0.2 = 0.18 \end{aligned}$$

$$\therefore V = \begin{bmatrix} 0.0944 & -0.2771 \\ 0.675 & 0.18 \end{bmatrix} \quad W = [0.299 \quad -0.4253]$$

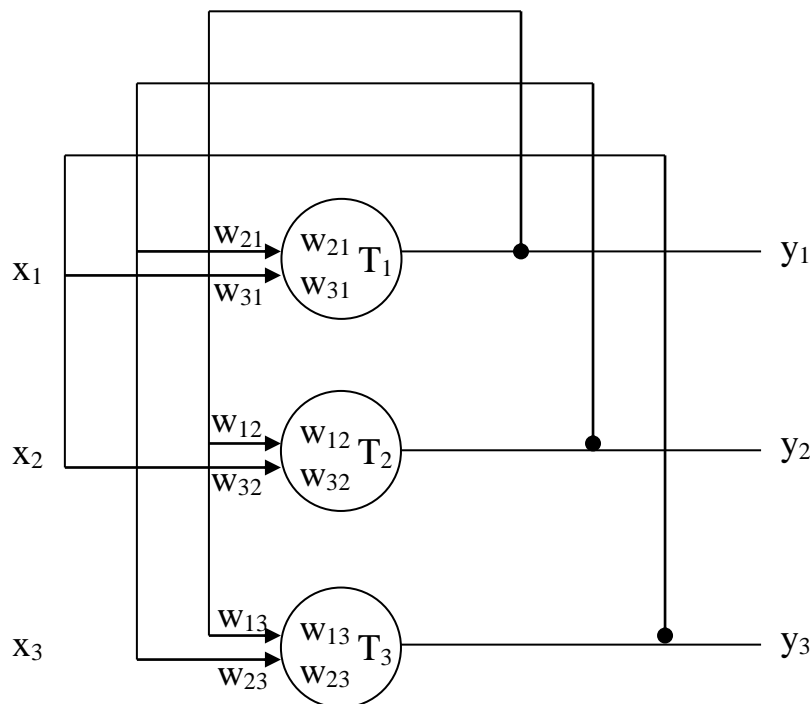
3.2 The Hopfield Network

The Nobel prize winner (in physics) John Hopfield has developed the discrete Hopfield net in (1982-1984). The net is a fully interconnected neural net, in the sense that each unit is connected to every other unit. The discrete Hopfield net has symmetric weights with no self-connections, i.e.,

$$W_{ij} = W_{ji}$$

$$\text{And } W_{ii} = 0$$

In this NN, inputs of 0 or 1 are usually used, but the weights are initially calculated after converting the inputs to -1 or +1 respectively.



“The Hopfield network“

The outputs of the Hopfield are connected to the inputs as shown in Figure, Thus feedback has been introduced into the network. The present output pattern is no longer solely dependent on the present inputs, but is also dependent on the previous outputs. Therefore the network can be said to have some sort of memory, also the Hopfield network has only one layer of neurons.

The response of an individual neuron in the network is given by :-

$$y_j = 1 \quad \text{if} \quad \sum_{i=1, i \neq j}^n W_{ij} X_i > T_j$$

$$y_j = 0 \quad \text{if} \quad \sum_{i=1, i \neq j}^n W_{ij} X_i < T_j$$

This means that for the j th neuron, the inputs from all other neurons are weighted and summed.

Note $i \neq j$, which means that the output of each neuron is connected to the input of every other neuron, but not to itself. The output is a hard-limiter which gives a 1 output if the weighted sum is greater than T_j and an output of 0 if the weighted sum is less than T_j . it will be assumed that the output does not change when the weighted sum is equal to T_j .

Thresholds also need to be calculated. This could be included in the matrix by assuming that there is an additional neuron, called neuron 0, which is permanently stuck at 1. All other neurons have input connections to this neuron's output with weight $W_{01}, W_{02}, W_{03}, \dots$ etc. this provides an offset which is added to the weighted sum. The relation ship between the offset and the threshold T_j is therefore:- $T_j = -W_{0j}$

The output $[y]$ is just the output of neuron 0 which is permanently stuck at 1, so

the formula becomes:- $[W_0] = [X]^t [Y_0]$

For example, if the patterns $X_1 = [0011]$ and $X_2 = [0101]$ are to be stored, first convert them to

$$X_1 = [-1 \quad -1 \quad 1 \quad 1]$$

$$X_2 = [-1 \quad 1 \quad -1 \quad 1]$$

To find the threshold:-

1- The matrix $\begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}$

2-The transpose of the matrix is
$$\begin{bmatrix} -1 & -1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix}$$

3- y_0 is permanently stuck at +1 , so the offsets are calculated as follows

$$W_0 = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ +1 & -1 \\ +1 & +1 \end{bmatrix} \begin{bmatrix} +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \\ 0 \\ +2 \end{bmatrix}$$

4-These weights could be converted to thresholds to give:-

$$T_1 = 2$$

$$T_2 = 0$$

$$T_3 = 0$$

$$T_4 = -2$$

$$T_j = -W_{0j}$$

EX7:-

Consider the following samples are stored in a net:-

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & +1 & -1 & -1 \\ +1 & +1 & -1 & -1 \\ -1 & -1 & +1 & +1 \end{bmatrix}$$

binary \rightarrow convert \rightarrow bipolar

The binary input is (1110). We want the net to know which of samples is the i/p near to?

Note :-

A binary Hopfield net can be used to determine whether an input vector is a “known” vector (i.e., one that was stored in the net) or “unknown” vector.

Solution:- 1-use Hebb rule to find the weights matrix

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{bmatrix}$$

$W_{ii}=0$ (diagonal)

$$W_{ij}=W_{ji} \quad \begin{matrix} & 1 & 2 & 3 & 4 \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{bmatrix} 0 & W_{12} & W_{13} & W_{14} \\ W_{21} & 0 & W_{23} & W_{24} \\ W_{31} & W_{32} & 0 & W_{34} \\ W_{41} & W_{42} & W_{43} & 0 \end{bmatrix} \end{matrix}$$

$$W_{12} = (-1*1) + (1*1) + (-1*-1) = 1$$

$$W_{13} = (-1*-1) + (1*-1) + (-1*1) = -1$$

$$W_{14} = (-1*-1) + (1*-1) + (-1*1) = -1$$

$$W_{21} = W_{12} = 1$$

$$W_{23} = (1*-1) + (1*-1) + (-1*1) = -3$$

$$W_{24} = (-1*-1) + (1*-1) + (-1*1) = -3$$

$$W_{31} = W_{32} = 1$$

$$W_{32} = W_{23} = -3$$

$$W_{34} = (-1*-1) + (-1*-1) + (1*1) = 3$$

$$W_{41} = W_{14} = -1$$

$$W_{42} = W_{24} = -3$$

$$W_{43} = W_{34} = 3$$

$$\therefore W = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -3 & -3 \\ -1 & -3 & 0 & 3 \\ -1 & -3 & 3 & 0 \end{bmatrix}$$

2-The i/p vector $x = (1 \ 1 \ 1 \ 0)$. For this vector, $y = (1 \ 1 \ 1 \ 0)$

Choose unit y_1 to update its activation

$$y - \text{in}1 = X_1 + \sum_j^m y_j w_{j1}$$

$$y - \text{in}1 = 1 + [(0*1) + (1*1) + (-1*1) + (-1*0)] \\ = 1 + 0 = 1$$

$$\therefore y = (1110)$$

Choose unit y_2 to up date its activation:-

$$y - \text{in}2 = x_2 + \sum_j y_j w_{j2} \\ = 1 + [(1*1) + (1*0) + (1*-3) + (0*-3)] \\ = 1 + (-2) = -1$$

$$y - \text{in}2 < 0 \quad \therefore y_2 = 0$$

$$\therefore y = (1010)$$

Choose unit y_3 to update its activation:-

$$y - \text{in}3 = x_3 + \sum_j y_j w_{j3} \\ = 1 + [(1*-1) + (1*-3) + (1*0) + (0*3)] \\ = 1 + (-4) = -3$$

$$y - \text{in}3 < 0 \quad \therefore y_3 = 0$$

$$\therefore y = (1000)$$

Choose unit y_4 to update its activation:-

$$y - \text{in}4 = x_4 + \sum_j y_j w_{j4} \\ = 0 + [(1*-1) + (1*-3) + (1*3) + (0*0)] \\ = 0 + (-1) = -1$$

$$y - \text{in}4 < 0 \quad y_4 = 0$$

$$y = (1000)$$

3- Test for convergence, false

\therefore The input vector $x = (1000)$, for this vector,

$$Y = (1 \ 0 \ 0 \ 0)$$

$$y - \text{in}1 = 1$$

$$y - \text{in}2 = 1$$

$$y - \text{in}3 = -1 = 0$$

$$y - \text{in}4 = -1 = 0$$

$$\therefore y = (1100)$$

\therefore The input vector $x = (1 \ 1 \ 0 \ 0)$

$$Y = (1 \ 1 \ 0 \ 0)$$

$$y - \text{in}1 = 2 = 1$$

$$y - \text{in}2 = 2 = 1$$

$$y - \text{in}3 = -4 = 0$$

$$y - \text{in}4 = -4 = 0$$

$$\therefore y = (1100)$$

The input is near to the second sample.

True.

Stop.

H.W

1-find the weights and thresholds for a Hopfield network that stores the patterns:- $(0 \ 0 \ 1)$ and $(0 \ 1 \ 1)$.

2-There are special techniques should be used to get out of local minima, explain it.

3.3 Bidirectional Associative Memory (BAM)

A bidirectional associative memory (BAM) is very similar to a Hopfield network, but has two layers of neurons (Kosko, 1988) and is fully connected from each layer to the other. There are feedback connections from the output layer to the input layer.

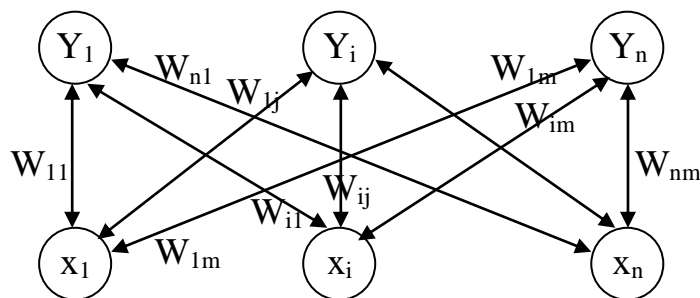
The BAM is hetero associative, that is, it accepts an input vector on one set of neurons and produces a related, but different, output vector on another set. The weights on the connections between any two given neurons from different layers are the same.

The matrix of weights for the connections from the output layer to the input layer is simply the transpose of the matrix of weights for the connections between the input and output layer.

Matrix for forward connection weights = w

Matrix for backward connection weights = w^T

There are 2 layers of neurons, an input layer and an output layer. There are no lateral connections, that is, no two neurons within the same layer are connected. Recurrent connections, which are feedback connections to a neuron from itself, may or not be present. Unlike the Hopfield network, the diagonal of the connection matrix is left intact, also the number of bits in the input pattern need not be the same as the output pattern, so the connection matrix is not necessarily square.



“ Layout of BAM Network “

The BAM operates by presenting on input pattern,[A], and passing it through the connection matrix to produce an output pattern,[B] .so:-

$$B[k] = f([A(k)][w])$$

Where

K: indicates time

A(k), B(k) :- are equivalent to [x] and [y]

F: activation function

W:- weight matrix between layer 1 & layer 2

The output of the neurons are produced by the function $f()$ which, like the Hopfield, is a hard-limiter with special case at θ .

This output function is defined as follows :-

$$\text{out}_i(k+1) = 1 \quad \text{if } \text{Net}_i(k) > 0$$

$$\text{out}_i(k+1) = 0 \quad \text{if } \text{Net}_i(k) < 0$$

$$\text{out}_i(k+1) = \text{out}_i(k) \quad \text{if } \text{Net}_i = 0 \text{ unchanged}$$

The output [B], is then passed back through the connection matrix to produce a new input pattern, [A].

$$A(k+1) = f([B(k)][W^T])$$

The [A] & [B] pattern are passed back and forth through the connection matrix in the way just described until there are no further changes to the values of [A] & [B]

محاسن الـ BAM

- 1- منسجمة مع الدوائر التناظرية والانظمة البصرية
- 2- لها اقتراب سريع في عملية التعلم والاسترجاع
- 3- لها حصانة ضد الـ noisy data
- 4- الاوزان ثابتة

مساوئ الـ BAM

- 1- سعة الخزن محددة
- 2- لها استجابة زائفة

3- احياناً تسلك سلوك غير متوقع

No learning -4

EX8:- let us try to train a network to remember three binary – vector pairs. A_i, B_i have the same number of component, using the Hebb rule to star :-

$$\begin{aligned} A_1 &= (1 \ 0 \ 0) & B_1 &= (0 \ 0 \ 1) \\ A_2 &= (0 \ 1 \ 0) & B_2 &= (0 \ 1 \ 0) \\ A_3 &= (0 \ 0 \ 1) & B_3 &= (1 \ 0 \ 0) \end{aligned}$$

1- Find the weight matrix?

2-Apply an input vector $A_1 = (1 \ 0 \ 0)$ to test the net to remember A_1 .**Sol**

$$W_1 = [A_1^T][B_1]$$

$$W_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$W_2 = [A_2^T][B_2]$$

$$W_2 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

$$W_3 = [A_3^T][B_3]$$

$$W_3 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$W = W_1 + W_2 + W_3$$

$$W = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \begin{bmatrix} -1 & 1 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix}$$

Test for A_1

$$A_1 * W = B_1$$

$$\begin{bmatrix} 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -3 & -3 & 5 \end{bmatrix}$$

$$\begin{bmatrix} -3 & -3 & 5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = B_1$$

Or طريقة مختصرة

$$W = [AT] [B]$$

$$W = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 3 \\ -1 & 3 & -1 \\ 3 & -1 & -1 \end{bmatrix}$$

And then continues the same steps .

H.W

Q1: find the weights and thresholds for a Hopfield network that stores the pattern 001 and 011.

Q2- A BAM is trained using the following input and output patterns:-

Input	Output
000010010000010	01
000010000010000	10
000100100100000	11

Find the weights that would be generated for the BAM network, and check that the input patterns generate the corresponding output patterns.

Q3- Briefly explain the following :-

1- Single layer network , Multi layer network

2-ANN

3- Areas of Neural network

4- supervised Learning , unsupervised Learning

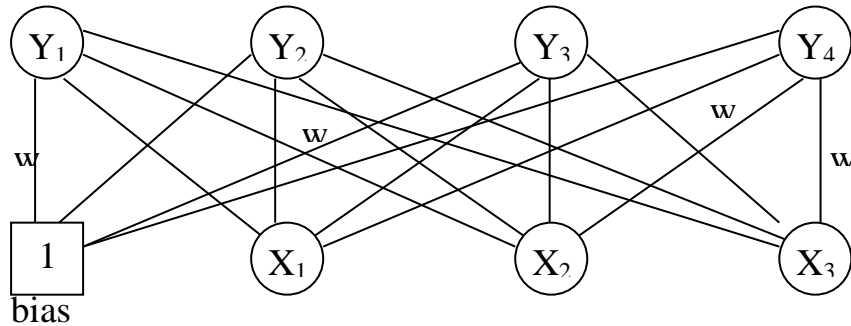
5-Recurrnt, non recurrent

6-Advantage & disadvantage of BAM

7- write the complete alg. Of BAM.

3.4 Adaline Neural Network

Adaline is the short form of "Adaptive linear neuron" and was presented in 1960 by B. Widrow and N. E. Hoff [WH1960]. The network is single layered, the binary values to be assumed for input and output are -1 and +1 respectively. Figure bellow shows the general topology of the network.



“Topology of Adaline “

Where

X = input vector (including bias)

Y = output vector = $f(w * X)$

W = weight matrix

An Adaline can be trained using the delta rule, also known as the least mean squares (LMS) or widerow- Holf rule. The learning rule minimize the mean squared error between the activation and the target value. This allows the net to continue learning on all training patterns, even after the correct output value is generated (if a threshold function is applied) for some patterns.

When the Adaline is in its tracing or learning phase, there are three factors to be taken into account

1-the inputs that are applied are chosen from a training set where the desired response of the system to these inputs is known.

2-the actual output produced when an input pattern is applied is compared with the desired output and used to calculate an error δ

3- the weight are adjusted to reduce the error .

This kind of training is called supervised learning because the output are known and the network is being forced into producing the correct outputs.

Three additional points need to be included before the learning rule can be used:-

4-the constant, η , has to be decided. The original suggestion for the Adaline was that η is made equal to:-

$$\eta = 1/(n + 1)$$

Where n is the number of inputs.

The effect of adjusting the weights by this amount is to reduce the error for the current input pattern to zero. In practice if η is set to this value the weights rarely settle down to a constant value and a smaller value is generally used.

5-the weight are initially set to a small random value. This is to ensure that the weights are all different.

6-the offset, w_0 gets adjusted in the same way as the other weights, except that the corresponding input x_0 is assumed to be +1.

The steps for solving any question in Adaline by using Delta-rule are :-

1-compute the learning coefficient η :-

$$\eta = 1/(n + 1)$$

n= number of inputs

2-comput neti :-

$$\text{net}_i = \sum_{i=1} x_i \cdot w_i$$

3-compute the error δ

$$\delta = d - \text{net}_i \quad d \text{ is the desired o/p}$$

4-compute the value of $\eta \cdot \delta \cdot x_i$ for all weights

5-find the total for all weight $\text{total} = \sum \eta \delta x_i$

6-find mean i $\text{mean } i = \text{total}/p$

Where :-

P:- is the no. of states

7- adjust the weights depending on mean i $W_i^{\text{new}} = W_i^{\text{old}} + \text{mean } i$

EX9 :-

Adaline is given the four different input and output combinations of the two input AND function, $y = x_1 \wedge \neg x_2$, as training set

$w_0 = -0.12$

$w_1 = 0.4$

$w_2 = 0.65$

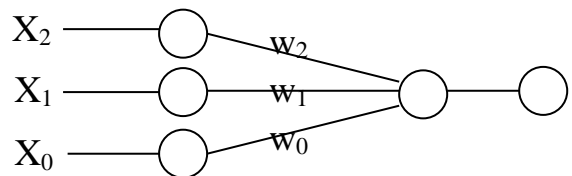
$y = x_1 \wedge \neg x_2$

X ₁	X ₂	Y
0	0	0
0	1	0
1	0	1
1	1	0

bias

X ₀	X ₁	X ₂	Y
+1	-1	-1	-1
+1	-1	+1	-1
+1	+1	-1	+1
+1	+1	+1	-1

Desired o/p



First the input pattern : +1 -1 -1

Weights : -0.12 0.4 0.65

$$\text{net} = \sum_{i=1}^n x_i \cdot w_i$$

$$= (+1 * -0.12) + (-1 * 0.4) + (-1 * 0.65) = -1.17 \text{ (actual output)}$$

d = desired output = -1 (for first pattern)

$$\therefore \delta = d - \text{net}$$

$$= -1 - (-1.17) = 0.17 \text{ (error)}$$

$$\eta = 0.1$$

Also we must compute :-

$$\Delta w_{ij} = \eta \cdot \delta \cdot x_i$$

For convenience , these figures have been rounded to two places after the decimal point, so become :- $\eta \cdot \delta \cdot x_0 = (0.1 * 0.17 * +1) = 0.017 \approx 0.02$

ملاحظة :- نستمر بالعمل مع بقية ال input وبهذا نحصل على النتائج التالية :-

								Δw_0	Δw_1	Δw_2
X ₀	X ₁	X ₂	W ₀	W ₁	W ₂	net	d	$\eta\delta x_0$	$\eta\delta x_1$	$\eta\delta x_2$
+1	-1	-1	-0.12	0.40	0.65	-1.17	-1	0.02	-0.02	-0.02
+1	-1	+1	0.12	0.40	0.65	0.13	-1	-0.11	0.11	-0.11
+1	+1	-1	-0.12	0.40	0.65	-0.37	+1	0.14	0.14	-0.14
+1	+1	+1	-0.12	0.40	0.65	0.93	-1	-0.19	-0.19	-0.19
total								-0.14	0.04	-0.46

$$\text{mean}_j = \text{total}(\Delta w_{ij}) / p \quad p = 4$$

$$\text{Mean}_0 = -0.14/4 = -0.035 = -0.04$$

$$\text{Mean}_1 = -0.04/4 = -0.01$$

$$\text{Mean}_2 = -0.46/4 = -0.115 = -0.12$$

$$\therefore W_{ij}^{\text{new}} = W_{ij}^{\text{old}} + \text{mean}_j$$

$$W_0^{\text{new}} = -0.12 + (-0.04) = -0.16$$

$$W_1^{\text{new}} = -0.40 + (0.01) = -0.41$$

$$W_2^{\text{new}} = -0.66 + (-0.12) = -0.53$$

Continue until $\eta\delta x = 0$

X ₀	X ₁	X ₂	W ₀	W ₁	W ₂	net	d	$\eta\delta x_0$	$\eta\delta x_1$	$\eta\delta x_2$
+1	-1	-1	-0.16	0.41	0.53	-1.10	-1	0.01	-0.01	-0.01
+1	-1	+1	-0.16	0.41	0.53	0.04	-1	-0.10	0.10	-0.10
+1	+1	-1	-0.16	0.41	0.53	-0.25	+1	0.13	0.13	-0.13
+1	+1	+1	-0.16	0.41	0.53	0.78	-1	-0.18	-0.18	-0.18
total								-0.14	0.04	-0.44
mean								-0.04	0.01	-0.11

								Δw_0	Δw_1	Δw_2
X_0	X_1	X_2	W_0	W_1	W_2	net	d	$\eta\delta x_0$	$\eta\delta x_1$	$\eta\delta x_2$
+1	-1	-1	-0.50	0.50	-0.50	-0.50	-1	-0.05	0.05	0.05
+1	-1	+1	-0.50	0.50	-0.50	-1.50	-1	0.05	-0.05	0.05
+1	+1	-1	-0.50	0.50	-0.50	0.50	+1	0.05	0.05	-0.5
+1	+1	+1	-0.50	0.50	-0.50	-0.50	-1	-0.05	-0.05	-0.5
total								0.00	0.00	0.00

The network has successfully found a set of weight that produces the correct outputs for all of the patterns.

H.W

Q1: A 2-input Adaline has the following set of weights $w_0 = 0.3$, $w_1 = -2.0$, $w_2 = 1.5$. When the input pattern is $x_0 = 1$, $x_1 = 1$, $x_2 = -1$

And the desired output is 1

a- what is the actual output?

b- what is the value of δ ?

c- Assuming that the weights are updated after each pattern and the value η is $1/n+1$, what are the new values for the weights?

d- using these new values of weights, what would the output be for the same input pattern?

Q2: with η set to 0.5, calculate the weights (to one decimal place) in the following example after one iteration through the set of training patterns.

a- updating after all the patterns are presented

b- updating after each pattern is presented

X_0	X_1	X_2	W_0	W_1	W_2	net	d	$\eta\delta x_0$	$\eta\delta x_1$	$\eta\delta x_2$
+1	-1	-1	-0.2	0.1	0.3	-0.6	+1	0.8	-0.8	-0.8
+1	-1	+1					+1			
+1	+1	-1					-1			
+1	+1	+1					+1			

3.5 Kohonen Network

Teuvo kohonen presented the self-organizing feature map in 1982. it is an unsupervised, competitive learning , clustering network in which only one neuron (or only one neuron in a group) is “on” at a time.

The self-organizing neural networks, also called (topology –preserving maps), assume a topological structure among the cluster units. This property is observed in the brain, but is not found in other artificial neural networks.

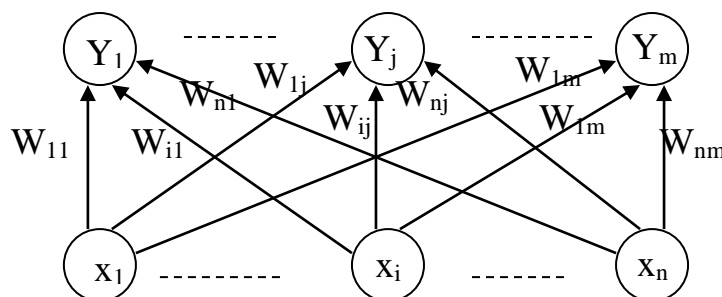
There are m cluster units arranged in a one –or two – dimensional array.

Cluster: وهي مجاميع من المعلومات كل مجموعة لها صفة معينة .

The weight vector for cluster units serves as an exemplar of the input patterns associated with that cluster. During the self organizing process, the cluster unit whose weight vector matches the input pattern most closely (typically, the square of the minimum Euclidean distance) is chosen as the winner. The winning unit and its neighboring units update their weights. The weight vectors of neighboring units are not, in general, close to the input pattern.

3.5.1 Architecture

A kohonen network has two layers, an input layer to receive the input and an output layer. Neurons in the output layer are usually arranged into a regular two dimensional array. The architecture of the kohonen self-organizing map is shown bellow.



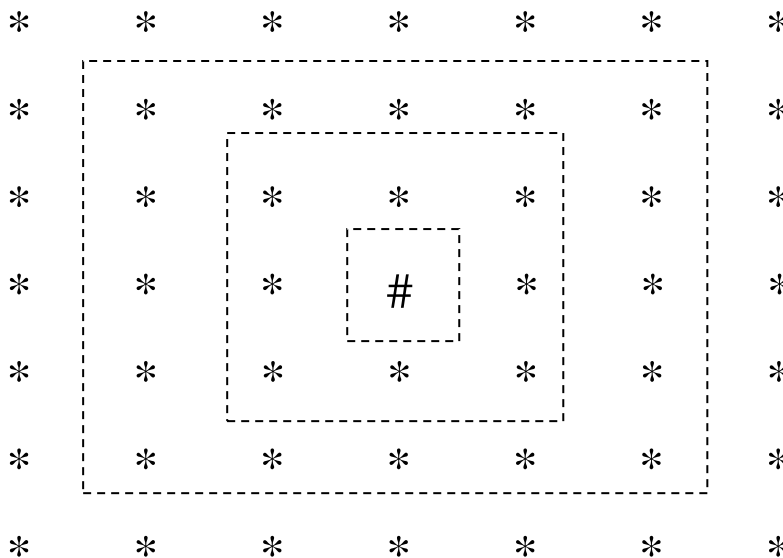
(kohonen self-organizing map)

$$* * * \left[* \left(* \left[\# \right] * \right) * \right] * *$$

$R_2 \quad R_1 \quad R_0 \quad R_1 \quad R_2$

Linear array 10 cluster

Neighborhoods of the unit designated by # of radii R=2 (1& 0) in a one – dimensional topology (with 10 cluster units) are shown in figure (4.2)



Neighborhoods for rectangular grid

R₀ =

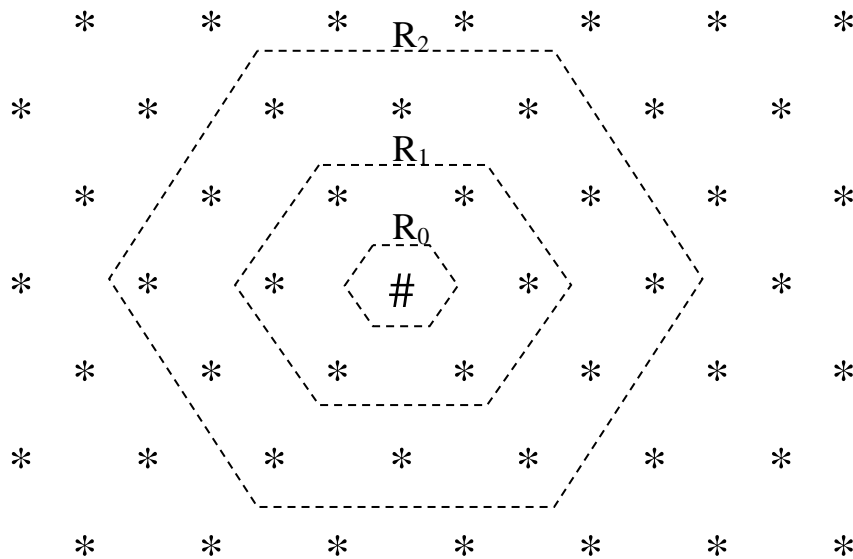
R₁ = _____

R₂ = - - - - -

* * * * *

The Neighborhoods of unit radii R=2 (1 & 0) are shown in figure (4.3) for a rectangular grid and in figure (4.4) for hexagonal grid (each with 49 units). In each illustration, the winning unit is indicated by the symbol “#” and the other units are denoted by “*” .

* Kohonen NN can be used in speech recognizer



Neighborhoods for hexagonal grid

$$R_0 = \dots\dots\dots$$

$$R_1 = \text{-----}$$

$$R_2 = \text{-----}$$

3.5.2 Algorithm

Step 0 : initialize weights w_{ij}

Set topological neighborhood parameters

Set Learning rate parameters.

Step1: while stopping condition is false, do step 2-8

Step2: for each input vector x , do step 3-5

Step3: for each j , compute distance

$$D(j) = \sum_i (x_i - w_{ij})^2 \quad \text{Euclidean distances}$$

Step4 : find index J such that $D(J)$ is a minimum

Step5: for all units j within a specified neighborhood of J , and for all i :

$$W_{ij}(\text{new}) = W_{ij}(\text{old}) + \alpha [X_i + W_{ij}(\text{old})]$$

Step6: update learning rate.

Step7: Reduce radius of topological neighborhood at specified times

Step8: Test stopping condition.

EX 10

A kohonen self-organizing map (SOM) to be cluster four vectors

$$\text{vector1} = (1 \ 1 \ 0 \ 0)$$

$$\text{vector2} = (0 \ 0 \ 0 \ 1)$$

$$\text{vector3} = (1 \ 0 \ 0 \ 0)$$

$$\text{vector4} = (0 \ 0 \ 1 \ 1)$$

The maximum no. of clusters to be formed is $m=2$ with learning rate $\alpha=0.6$

Sol:

With only 2 clusters available, the neighborhood of node J is set so that only one cluster updates its weight at each step

Initial weight matrix:

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

1- for the first vector

x_1	x_2	x_3	x_4
(1	1	0	0)

$$D(1) = (1-0.2)^2 + (1-0.6)^2 + (0-0.5)^2 + (0-0.9)^2 = 1.86$$

$$D(2) = (1-0.8)^2 + (1-0.4)^2 + (0-0.7)^2 + (0-0.3)^2 = 0.98 \text{ (Minimum)}$$

$\therefore J = 2$ (The input vector) is closest to output node 2)

\therefore The weight on the winning unit is update:-

$$\begin{aligned} W_{21}(\text{new}) &= W_{12}(\text{old}) + 0.6(x_i - W_{12}(\text{old})) \\ &= 0.8 + 0.6(1 - 0.8) = 0.92 \end{aligned}$$

$$\begin{aligned} W_{22}(\text{new}) &= 0.4 + 0.6(1 - 0.4) \\ &= 0.4 + 0.36 = 0.76 \end{aligned}$$

$$W_{23}(\text{new}) = 0.7 + 0.6(0 - 0.7) \\ = 0.28$$

$$W_{24}(\text{new}) = 0.3 + 0.6(0 - 0.3) \\ = 0.12$$

This gives the weight matrix

$$\begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

2- for the second vector (0 0 0 1)

$$D(i) = (0 - 0.2)^2 + (0 - 0.6)^2 + (0 - 0.5)^2 + (1 - 0.9)^2 = 0.66 \text{ minimum}$$

$$D(2) = (0 - 0.92)^2 + (0 - 0.76)^2 + (0 - 0.28)^2 + (1 - 0.12)^2 = 2.2768$$

∴ J = 1 (The i/p vector is closest to o/p node 1)

After update the first column of the weight matrix:-

$$\begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

3- for the third vector (1 0 0 0)

$$D(i) = (-0.08)^2 + (0 - 0.24)^2 + (0 - 0.20)^2 + (0 - 0.96)^2 = 1.856$$

$$D(2) = (1 - 0.92)^2 + (0 - 0.76)^2 + (0 - 0.28)^2 + (1 - 0.12)^2 \\ = 2.2768 \text{ minimum}$$

∴ J = 2 (The i/p vector is closest to o/p node (2))

After update the second column of the weight matrix:-

$$\begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

4- for the fourth vector (0 0 1 1)

$$D(i) = (0 - 0.08)^2 + (0 - 0.24)^2 + (1 - 0.20)^2 + (1 - 0.96)^2 = 0.7056 \text{ minimum}$$

$$D(2) = (0 - 0.968)^2 + (0 - 0.304)^2 + (1 - 0.112)^2 + (1 - 0.048)^2 = 2.724$$

$\therefore J = 1$ (the i/p vector is closest to o/p node 1)

After update the first column of the weight matrix :-

$$\begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

\therefore Reduce the learning rate

$$\alpha(t+1) * \alpha(t) = 0.5 * (0.6) = 0.3$$

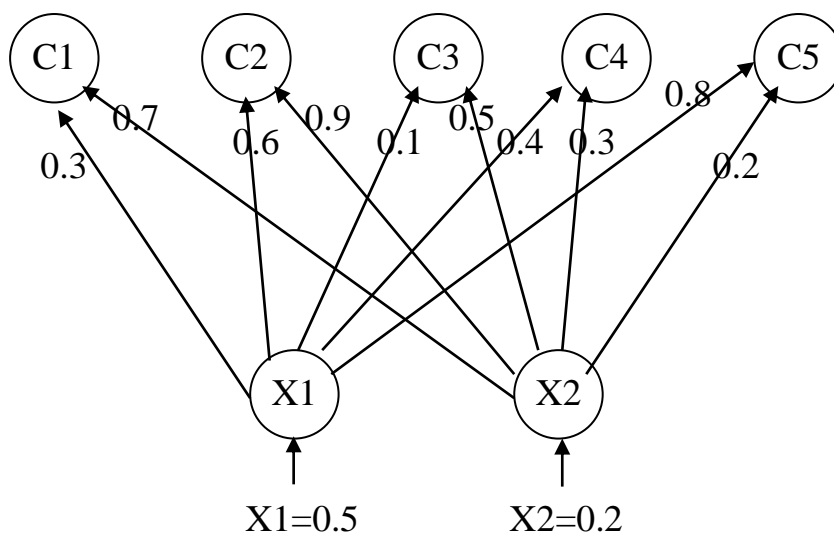
\therefore After one iteration the weight matrix will be:-

$$\begin{bmatrix} 0.032 & 0.970 \\ 0.096 & 0.300 \\ 0.680 & 0.110 \\ 0.984 & 0.048 \end{bmatrix}$$

H.W

Find the output node with minimum distance then update its reference vector

only $\alpha = 0.5$



3.6 Self- Organizing Networks

Self –organizing networks mean that the systems are trained by showing examples of patterns that are to be classified, and the network is allowed to produce its own output code for the classification.

In self – organizing networks the training can be supervised or unsupervised. The advantage of unsupervised learning is that the network finds its own energy minima and therefore tends to be more efficient in terms of the number of patterns that it can accurately store and recall.

In self – organizing networks four properties are required:-

- 1- The weight in the neurons should be representative of a class of patterns.
So each neuron represents a different class
- 2- Input patterns are presented to all of the neurons, and each neuron produces an output. The value of the output of each neuron is used as a measure of the match between the input pattern and the pattern stored in the neuron
- 3- A competitive learning strategy which selects the neuron with the largest response.
- 4- A method of reinforcing the largest response.

3.7 Adaptive Resonance Theory (ART)

Adaptive resonance theory (ART) was developed by Carpenter and Grossberg (1987). One form, ART 1, is designed for clustering binary vectors, another, ART2 also by Carpenter and Grossberg (1987).

These nets cluster inputs by using unsupervised learning input patterns may be presented in any order. Each time a pattern is presented, an appropriate cluster unit is chosen and that cluster's weights are adjusted to let the cluster unit learn the pattern.

3.7.1 Basic Architecture

Adaptive resonance theory nets are designed to allow the user to control the degree of similarity of patterns placed on the same cluster. ART1 is designed to cluster binary input vectors. The architecture of an ART1 net Consists of the following units:-

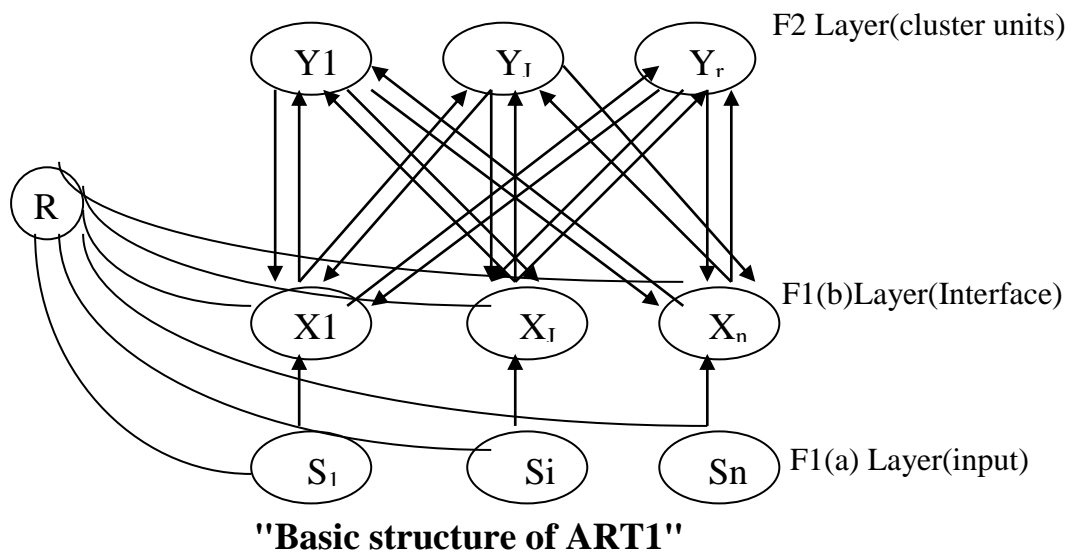
- 1- Computational units.
- 2- Supplemental units.

1- Computational units:-

The architecture of the computational units for ART1 consists of three field of unites:-

- 1- The F1 units (input and interface units)
- 2- The F2 units (cluster units)
- 3- Reset unite

This main portion of the ART1 architecture is illustrated in figure bellow:-



The F1 layer can be considered to consist to two of two parts:-

- 1- F1 (a) the input units
- 2- F1 (b) the interface units.

Each unit in the F1 (a) (input) layer is connected to the corresponding unit in the F1 (b) (interface) layer .Each unit in the F1 (a) &F1 (b) layer is connected to the reset unit, which in turn is connected to every F2 unit. Each unit in the F1 (b) is connected to each unit in the F2 (cluster) by two weighted pathways:-

1- Bottom –up weights:-

The F1(b) unit X_i is connected to the F2 unit Y_j by bottom –up weights b_{ij} .

2- Top- down weights:-

Unit Y_j is connected to unit X_i by top-down weights t_{ji} .

The F2 layer is a competitive layer in which only the uninhibited node with the largest net input has a non zero activation.

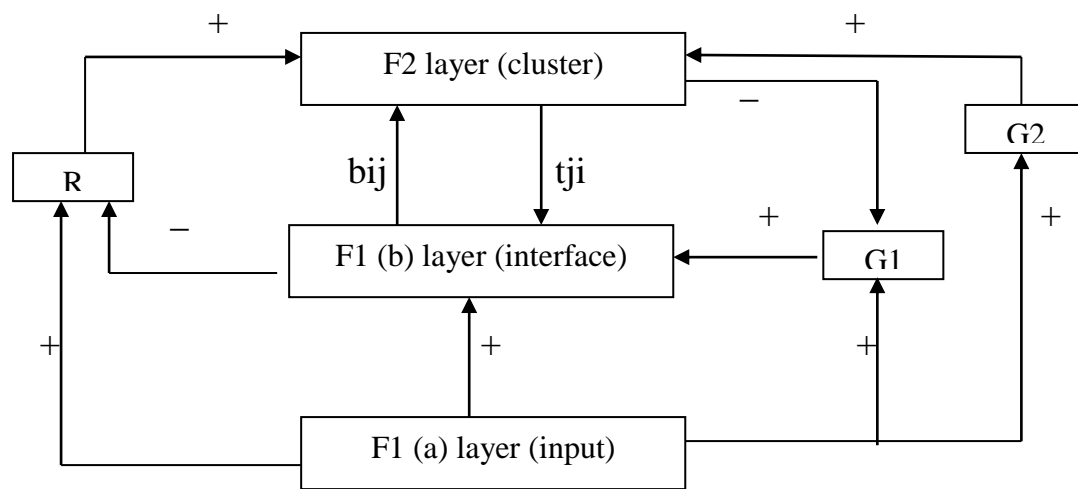
2-Supplemental Units:-

The Supplemental Units shown in figure (4-6) are important from a theoretical point of view. There are two Supplemental Units called gain control units, these are:-

1- Gain1 \longrightarrow g_1 or G_1

2- Gain2 \longrightarrow G_2

In addition to the reset unit \underline{R}

**“The Supplemental Units for ART1”**

Excitatory signals are indicated by (+) and inhibitory signals by (-), a signal is sent whenever any unit in the designated layer is (on).

Each unit in either the F1 (b) or F2 layer of the ART1 net has three sources from which it can receive a signal

1- F1(b) can receive signals from :-

- F1(a) (an input signal)
- F2 node (top –down signal)
- G_1 unit.

2- F2 unit can receive a signal from :-

- F1 (b) (an interface unit)
- R unit (reset unit)
- G_2 unit

An F1(b) (interface) or F2 unit must receive two excitatory signals in order to be (on). Since there are three possible sources of signals, this requirement is called the two-thirds rule.

The reset unit R controls the vigilance matching (the degree of similarity required for patterns to be assigned to the same cluster unit is controlled by a user – specified parameter, known as the vigilance parameter). When any unit in the F1 (a) is on, an excitatory signal is sent to R. the strength of that signal depends on how many F1(a) are (on). R also receives inhibitory signals from the F1(b) that are (on). If enough F1(b) are (on), unit R is prevented from firing . If unit R does fire, it inhibits any F2 unit that is (on). This forces the F2 layer to choose a new winning node.

There are two types of learning that differ both in their theoretical assumptions and in their performance characteristics can be used for ART nets:-

fast learning

It is assumed that weight updates during resonance occur rapidly, in fast learning, the weight reach equilibrium on each trial. It is assumed that the ART1net is being operated in the fast learning mode.

slow learning

The weight changes occur slowly relative to the duration of a learning trial, the weights do not reach equilibrium on a particular trail.

H.W

Q1: Write the complete algorithm for kohonen neural network?

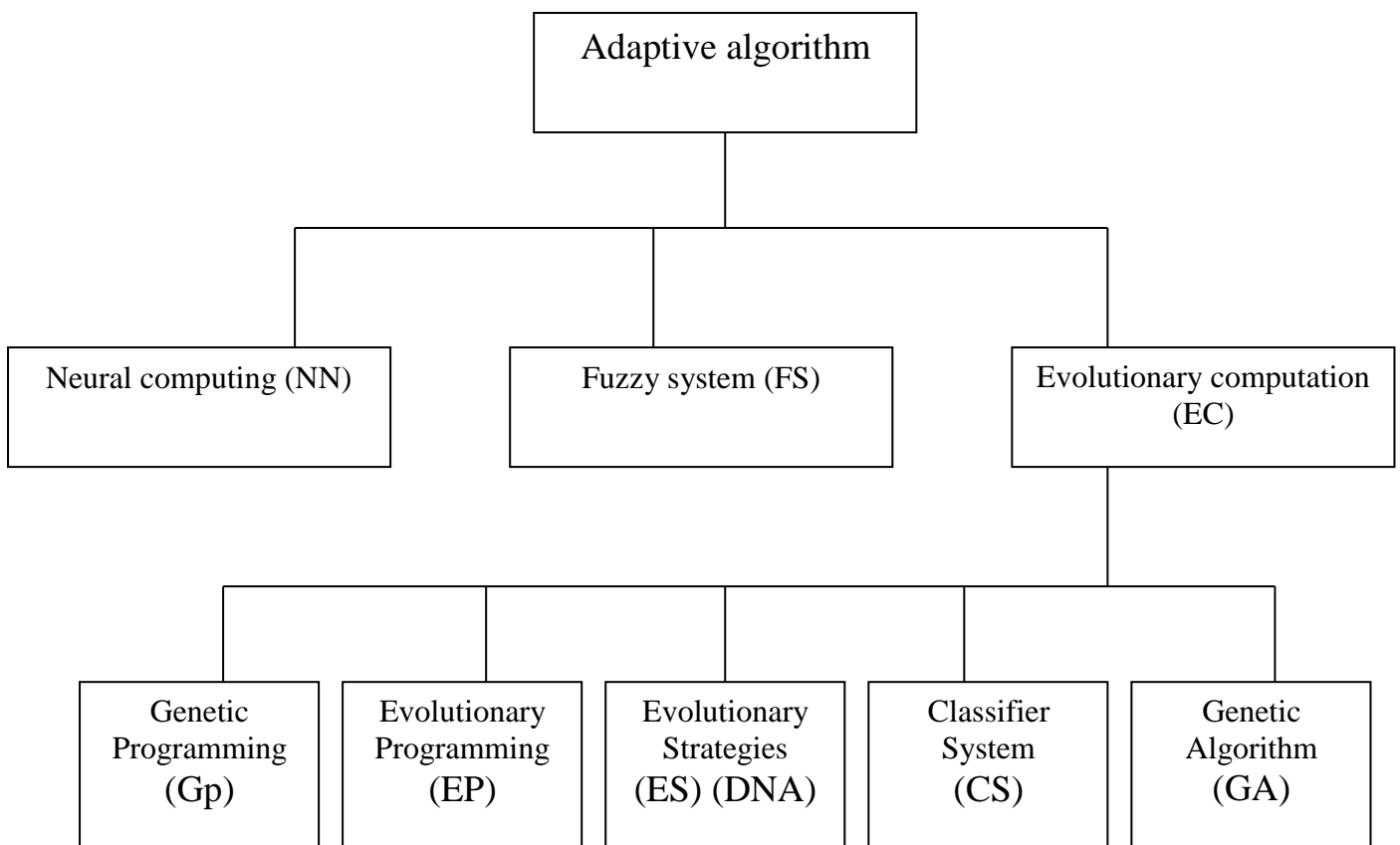
Q2: there are two basic units in ART1 architecture, list them and draw the figure for each one of them.

Q3: there are two kinds of learning in ART neural network. Briefly explain each one of them. Which kind does ART1 use?

Q4: Define the following expressions:-

- 1- Euclidean Distances
- 2- Vigilance matching
- 3- Bottom –up and top- down weights
- 4- Two-thirds rule

4.1 Genetic Algorithms (GA)



Structure of Adaptive Algorithm

A genetic algorithm is a search procedure modelled on the mechanics of natural selection rather than a simulated reasoning process. Domain Knowledge is embedded in the abstract representation of a candidate solution termed an organism. Organisms are grouped into sets called populations. Successive population are called generation. The aim of GA is search for goal.

A generational GA creates an initial generation $G(0)$, and for each generation , $G(t)$, generates a new one , $G(t+1)$. An abstract view of the algorithm is:-

Generate initial population, $G(0)$;

Evaluate $G(0)$;

$t:=0$;

Repeat

$t:=t+ 1$

Generate $G(t)$ using $G(t-1)$;

Evaluate $G(t)$;

Until solution is found.

4.1.1 Genetic Operators

The process of evolving a solution to a problem involves a number of operations that are loosely modeled on their counterparts from genetics .

Modeled after the processes of biological genetics , pairs of vectors in the population are allowed to “ mate” with a probability that is proportional to their fitness . the mating procedure typically involves one or more genetic operators .

The most commonly applied genetic operators are :-

- 1- Crossover.
- 2- Mutation.
- 3- Reproduction.

1- Crossover

Is the process where information from two parents is combined to form children. It takes two chromosomes and swaps all genes residing after a randomly selected crossover point to produce new chromosomes.

This operator does not add new genetic information to the population chromosomes but manipulates the genetic information already present in the mating pool (MP).

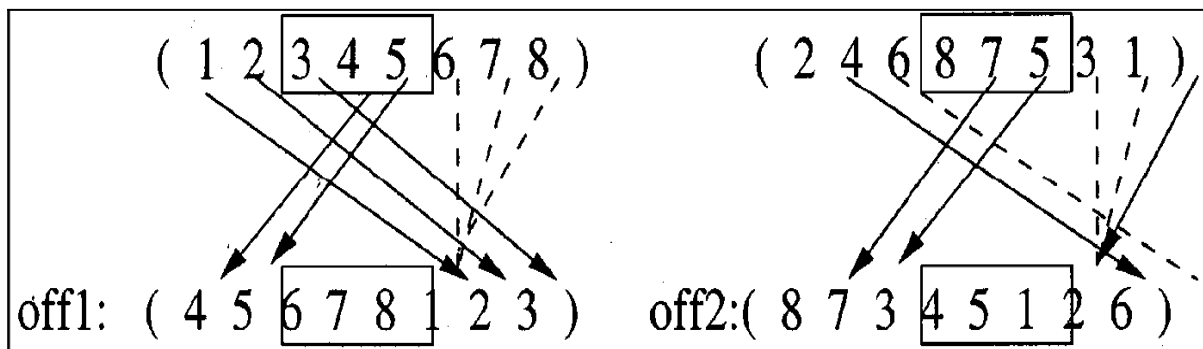
The hope is to obtain new more fit children It works as follows :-

- 1- Select two parents from the MP (The best two chromosomes) .
- 2- Find a position K between two genes randomly in the range (1, M-1)
M = length of chromosome
- 3- Swap the genes after K between the two parents.

The output will be the both children or the more fit one.

1- a Order crossover (OX1)

The *order crossover operator* (Figure 4) was proposed by Davis (1985). The OX1 exploits a property of the path representation, that the order of cities (not their positions) are important. It constructs an offspring by choosing a sub tour of one parent and preserving the relative order of cities of the other parent.



Order crossover (OX1)

For example, consider the following two parent tours:

(1 2 3 4 5 6 7 8) and
(2 4 6 8 7 5 3 1),

and suppose that we select a first cut point between the second and the third bit and a second one between the fifth and the sixth bit. Hence,

(1 2|3 4 5|6 7 8) and
(2 4|6 8 7|5 3 1).

The offspring are created in the following way. First, the four segments between the cut point are copied into the offspring, which gives

$$(**|3\ 4\ 5|***) \text{ and}$$

$$(**|6\ 8\ 7|***).$$

Next, starting from the second cut point of one parent, the rest of the cities are copied in the order in which they appear in the other parent, also starting from the second cut point and omitting the cities that are already present. When the end of the parent string is reached, we continue from its first position. In our example this gives the following children:

$$(8\ 7|3\ 4\ 5|1\ 2\ 6) \text{ and}$$

$$(4\ 5|6\ 8\ 7|1\ 2\ 3),$$

- *Partially mapped crossover (PMX)*
- *Cycle crossover (CX)*
- *Order based crossover (OX2)*
- *Position based crossover (POS)*
- *Heuristic crossover*
- *Genetic edge recombination crossover (ER)*
- *Sorted match crossover*
- *Maximal preservative crossover (MPX)*
- *Voting recombination crossover (VR)*
- *Alternating position Crossover (AP)*

2- Mutation

The basic idea of it is to add new genetic information to chromosomes. It is important when the chromosomes are similar and the GA may be getting stuck in Local maxima. A way to introduce new information is by changing the allele of some genes. Mutation can be applied to :-

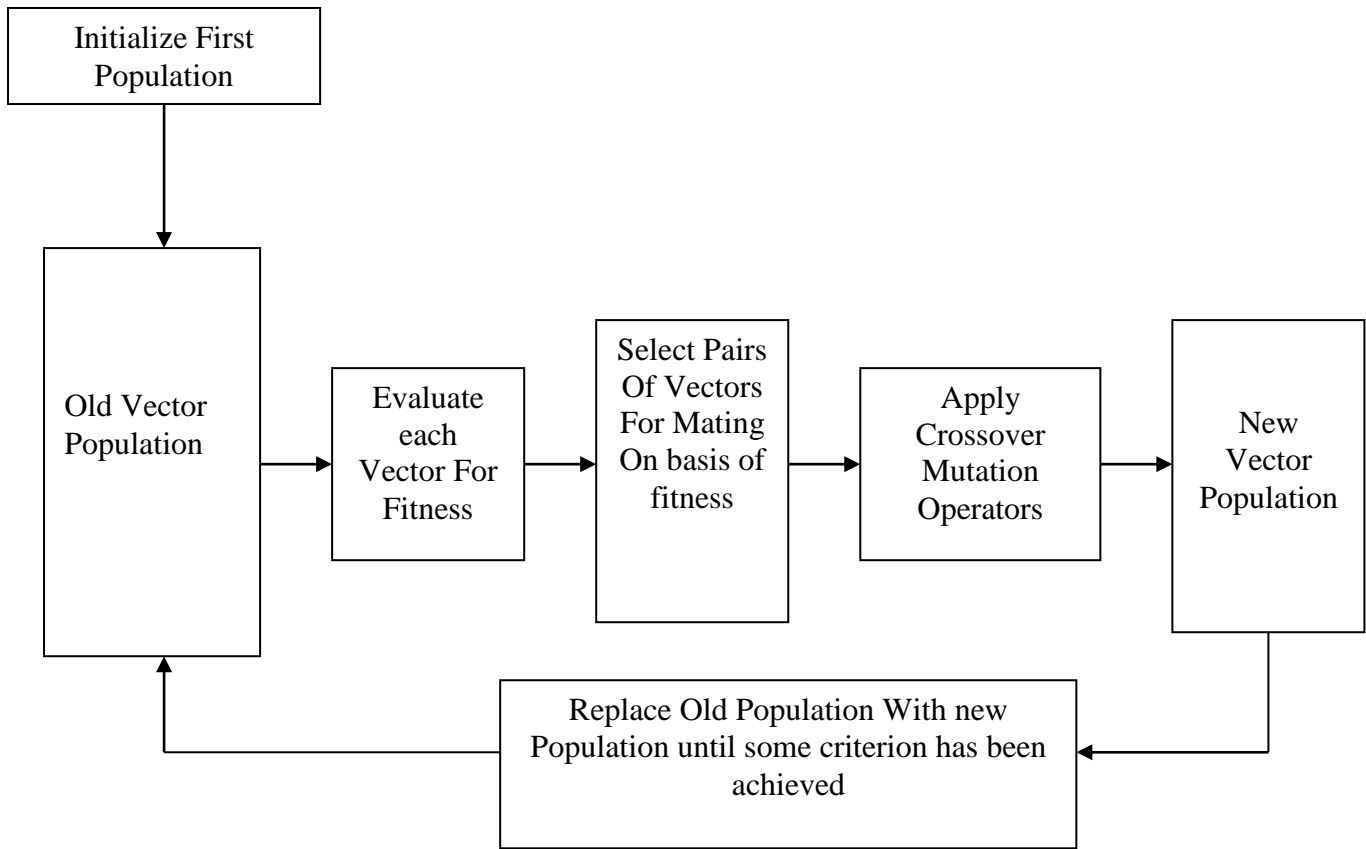
- 1- Chromosomes selected from the MP.
- 2- Chromosomes that have already subject to crossover.

The Figure bellow illustrates schematically the GA approach.

3- Reproduction

After manipulating the genetic information already present in the MP by fitness function the reproduction operator add new genetic information to the population of the chromosomes by combining strong parents with strong children , the hope is to obtain new more fit children . Reproduction imitate to the natural selection.

This schematic diagram of a genetic algorithm shows the functions that are carried out in each generation. Over a number of such generation the initial population is evolved to the point where it can meet some criterion with respect the problem at hand .



“Genetic Algorithm approach “

4.2 Genetic Programming (GP)

Genetic programming (GP) is a domain – independent problem – solving approach in which computer programs are evolved to solve, or approximately solve problems. Thus, it addresses one of the central goals of computer science namely automatic programming. The goal of automatic programming is to create, in an automated way, a computer program that enables a computer to solve a problem.

GP is based on reproduction and survival of the fittest genetic operations such as crossover and mutation. Genetic operation are used to create new offspring population of individual computer programs from the current population of programs .

GP has several properties that make it more suitable than other paradigms (e.g. . best – first search , heuristic search , hill climbing etc .) , these properties are :-

- 1- GP produces a solution to a problem as a computer program. Thus GP is automatic programming.
- 2- Adaptation in GP is general hierarchical computer programs of dynamically varying size & shape.
- 3- It is probabilistic algorithm.
- 4- Another important feature of GP is role of pre processing of inputs and post processing of outputs .

To summarize, genetic programming includes six components, many very similar to the requirements fo GAs:

- 1- A set of structures that undergo transformation by genetic operators.
- 2- A set of initial structures suited to a problem domain.
- 3- A fitness measure, again domain dependent, to evaluate structures.
- 4- A set of genetic operators to transform structures.

- 5- Parameters and state descriptions that describe members of each generation.
- 6- A set of termination conditions.

EX. 11:-

By using GA step by step, find the maximum number in 0 to 31. let k=3 and population size=4, and the initial population is:-

14	→	01110	}	population
3	→	00011		
25	→	11001		
21	→	10101		

Fitness function will be:-

25&21

3&14

25&21

1 1		0 0 1	→	25
1 0		1 0 1	→	21

1 1 1 0 1	→	29
1 0 0 0 1	→	17

14&3

0 1		1 1 0	→	14
0 0		0 1 1	→	3

0 1 0 1 1	→	11
0 0 1 1 0	→	6

the new population will be an array and we choose position [16] randomly to do mutation on it:-

1	1	1	0	1
1	0	0	0	1
0	1	0	1	1
0	0	1	1	0

Mutation →

1	1	1	0	1
1	0	0	0	1
0	1	0	1	1
1	0	1	1	0

Mutation 0 → 1

:

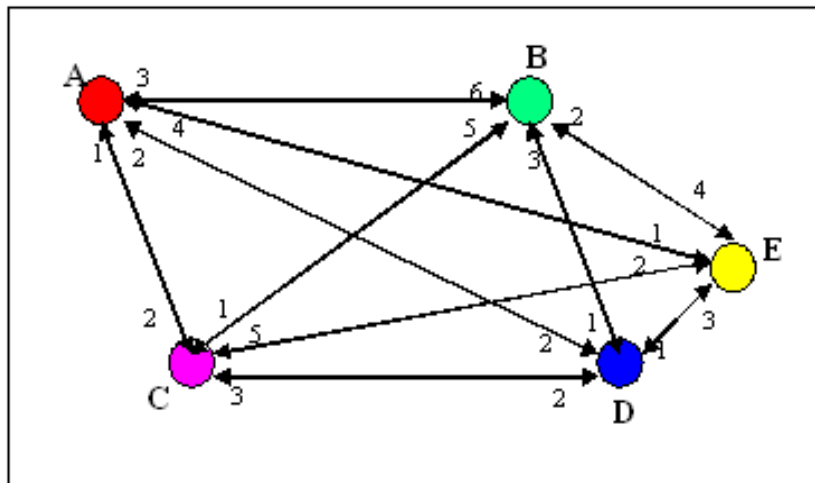
After Mutation the new population will be:

$$\begin{array}{l}
 1\ 1\ 1\ 0\ 1 = 29 \\
 1\ 1\ 0\ 0\ 1 = 25 \\
 1\ 0\ 1\ 0\ 1 = 21 \\
 1\ 0\ 1\ 1\ 0 = 22
 \end{array}
 \left. \vphantom{\begin{array}{l} 1\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 0\ 1 \\ 1\ 0\ 1\ 0\ 1 \\ 1\ 0\ 1\ 1\ 0 \end{array}} \right\} \text{reproduction}$$

Because the mutation we replace 17 with 22 in the new population.

EX 12

Apply GA in travelling salesman to find the shortest path . let k=2 and the initial population is:-



$$\begin{array}{l}
 A\ B\ C\ D\ E = 12 \\
 B\ C\ D\ E\ A = 10 \\
 A\ C\ D\ B\ E = 11 \\
 E\ C\ A\ D\ B = 11 \\
 B\ A\ D\ C\ E = 10 \\
 D\ E\ B\ A\ C = 10
 \end{array}
 \left. \vphantom{\begin{array}{l} A\ B\ C\ D\ E \\ B\ C\ D\ E\ A \\ A\ C\ D\ B\ E \\ E\ C\ A\ D\ B \\ B\ A\ D\ C\ E \\ D\ E\ B\ A\ C \end{array}} \right\} \text{initial population}$$

$$\begin{array}{l}
 B\ C\ D \mid E\ A = 10 \\
 B\ A\ D \mid C\ E = 10
 \end{array}$$

$$\begin{array}{l}
 B\ C\ D \quad A\ E = 6 \\
 B\ A\ D \quad E\ C = 13
 \end{array}$$

D E B	A C =10
A C D	B E

D E B	A C = 10
A C D	E B =9

E C A	D B =11
A B C	D E =12

E C A	D B = 11
A B C	D E =12

THE NEW POPULATION IS:-

B C D A E = 6
 B A D E C =13
 D E B A C = 10
 A C D E B = 9
 E C A D B =11
 A B C D E =12

أما نأخذ افضل الكروموسومات من الجيل الجديد و افضل الكروموسومات من الجيل السابق (بنفس عدد المجتمع) ونعمل crossover لهما او نعمل crossover للجيل الجيد فقط

H.W

Q1: Can the bit string 0 1 0 1 0 1 0 1 be the result of crossing over the following pairs of parents?:-

a- 11111111 and 00000000

b-01010101 and 11111111

c-10100101 and 01011010

Q2: What is genetic algorithm (GA). Explain its algorithm.

Q3: What are the most commonly operators used in GA, list it only, then draw the figure which illustrates schematically the GA approach.

Q4: Adaptive algorithm includes GA and GP in one port of it. Illustrates schematically the main structure of adaptive algorithm.

Web programming

- 1- Introduction web programming
- 2- HTML
- 3- JavaScript
- 4- ASP

Introduction

- Just as there is a diversity of programming languages available and suitable for conventional programming tasks, there is a diversity of languages available and suitable for Web programming.
- The Internet becomes the main method in exchanging cultures and transferring knowledge between people.

- The Web was originally designed to deliver static Web pages from a Web server connected somewhere on the Internet to a Web browser sitting on a user's desktop computer. Basically, all a user could do was click on a hot spot or hypertext link to retrieve a new page, read it, and then go on to the next page.
- The Web was not designed to support EC sites, especially B2C sites. In its original state, it was not possible to create pages that would allow consumers to easily determine what products were for sale, to select products as they moved from page to page (i.e., an electronic shopping cart), to place an order, or to verify an order. Similarly, there was no simple way to integrate a Web server with a database system containing product, pricing, The Web was originally designed to deliver

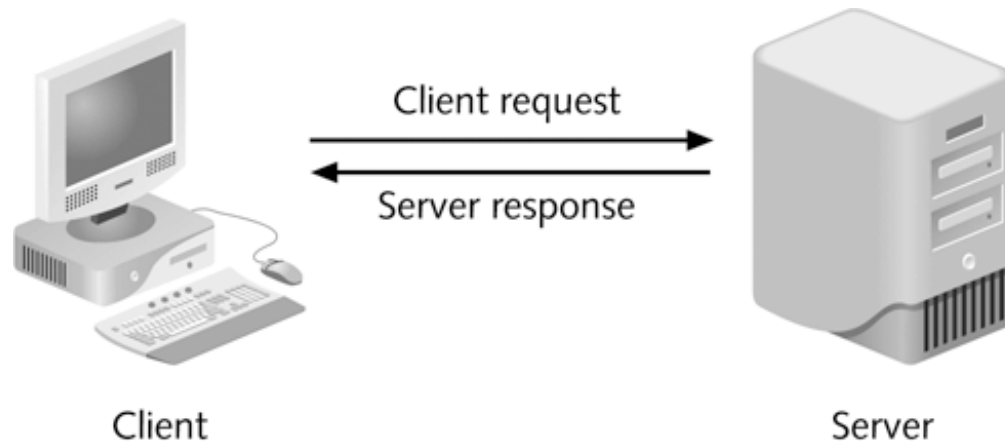
- and promotional data with transactional systems for processing orders and with payment systems for handling credit card purchases and settlements. Over time, these limitations have been addressed. First, forms were added to HTML. Forms provided a way to produce Web pages from which a consumer could select, order, and pay for products. Second, special programming and scripting languages (e.g., Java and JavaScript) were created. These newer languages allowed application developers to produce interactive Web pages whose functionality emulated the rich functionality of standard Windows-based applications. Finally, a standard application programming interface (API), called the common gateway interface (CGI), was introduced. Generally speaking, an API provides a way for one software program to communicate with another,

- whereas CGI provides a way for software developers and application programmers to integrate Web servers with various back-end programs and data sources.
- Because of CGI's inefficiencies, newer APIs and special database gateway programs were also introduced. As a result of these changes, the Web is now well suited for the dynamic world of EC.
- This appendix examines issues of end-user interactivity and dynamic data access. The first sections focus on Java and JavaScript, which are special programming languages that can be used to create Web pages with rich graphical user interfaces (GUIs). The remaining sections examine various methods—CGI programming, specialized APIs, and server-side scripting—for integrating a Web server with back-end programs, including relational databases.

Internet

- The Internet is a computer network made up of thousands of networks worldwide.
- All computers on the Internet communicate with one another using the Transmission Control Protocol/Internet Protocol suite, abbreviated to TCP/IP.
- Computers on the Internet use a client/server architecture.
- This means that the remote server machine provides files and services to the user's local client machine.
- Software can be installed on a client computer to take advantage of the latest access technology.

An Internet user has access to a wide variety of services :
electronic mail, file transfer, vast information resources,
interest group membership, interactive collaboration,
multimedia displays, real-time broadcasting



Web server

- A Web server is a computer that runs special serving software. That software "serves" HTML pages and the files associated with those pages when requested by a client, usually a Web browser.

Client (“front end”):

- Presents an interface to the user gathers information from the user, submits it to a server, then receives, formats, and presents the results returned from the server

Uniform Resource Locator URL

- A URL is a Web Page's address and identifies where the web page is stored on the Internet.
- It is a four-part addressing scheme.

Structure of a URL

Figure 6

The diagram shows the URL `http://www.centralpenn.edu/library/index.html` with four red curly brackets underneath it. The brackets are labeled as follows: 'protocol' above 'http://', 'server address' below 'www.centralpenn.edu', 'pathname' above 'library/index.html', and 'filename' below 'index.html'.

protocol
http://www.centralpenn.edu/library/index.html
server address filename

Internet Service Provider (ISP):

- Provides access to the Internet along with other types of services such as e-mail.

HyperText Transfer Protocol (HTTP)

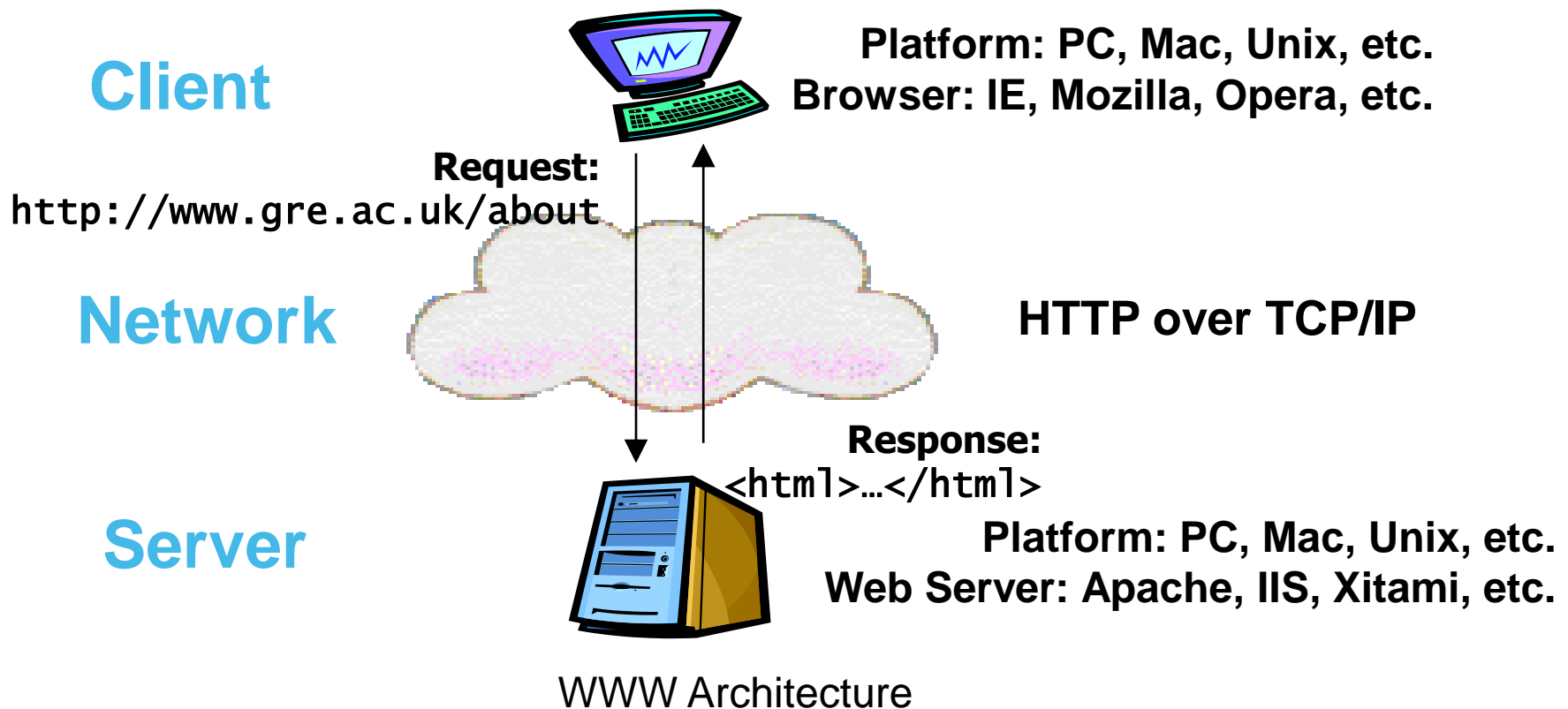
- to transmit data Protocols for other Internet applications

- Client-side:
 - [HTML / XHTML](#) (Extensible HyperText Markup Language)
 - [JavaScript / VBScript](#) (client-side scripting)
 - [Applets / ActiveX controls](#)
- Server-side:
 - [JSP](#) (Java Server Pages)
 - [ASP](#) (Active Server Pages)
 - [ASP.NET](#) (next generation of ASP)
 - [PHP](#)
 - [Python](#)

2- Web application

- An application which is accessed via web browser over a network is called web application or web based application. All the websites are examples of web applications.
- Web application is written in a server side scripting language like ASP (active server pages).
- When user types address of website for example www.programming-web.com, browser transmits request to the web server which hosts the required site.
- On web server, web server software like email receives this request and processes it

- The output generated by web server includes only those scripts which can be rendered by web browser.
- Above process repeats when user performs an action which requires communication with server such as submitting entry form or viewing another page.

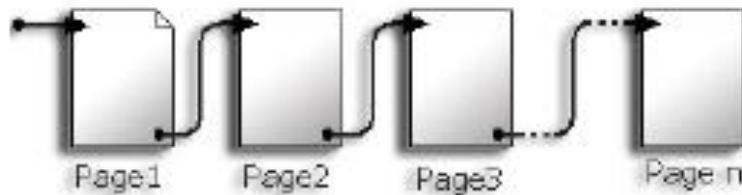


The Web Concepts

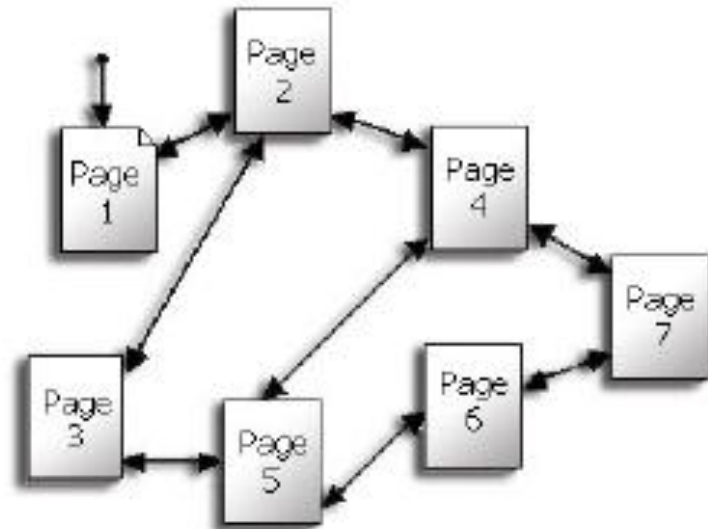
- The World-Wide Web (W3) was developed to be a pool of human knowledge, and human culture, which would allow collaborators in remote sites to share their ideas and all aspects of a common project.
- There are many Web concepts as following:

Hypertext:

- Hypertext links allow the reader to jump instantly from one electronic document to another.
- Two type : linear text and nonlinear text



Linear Text



Nonlinear Text -- Hypertext --

three basic “rules” of hypertext:

- A large body of information is organized into numerous fragments, or in the case of the Web, into pages.
- The pages relate to each other.
- The user needs only a small fraction of the information at any given moment.

Web Page

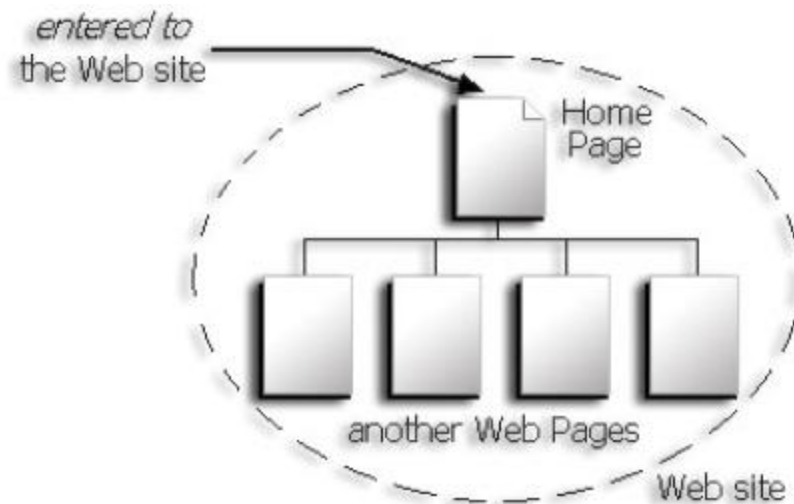
- The Web page is a space of information on the Internet, that presents information about a particular person, business, or organization or cause.
- The Web consists of files, called Web pages (documents).
- it is containing links to resources (text, images, audios, videos, and other data), throughout the Internet

Web Site

- A Web site is a group of related Web pages
- which presents information about a particular person, business, organization or cause
- A well-designed Web site is a collection of related Web documents (pages) that share a common theme, look, and feel.
- The first thing we can see when ‘enter’ the site is ‘Home Page’, that offers links to more detailed information on the different topics in the same subject covered by the site.

Web Browsing

- A web browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web.
- An information resource is identified by a Uniform Resource Identifier (URI) and may be a web page ‘image, video, or other piece.



- CGI :(common gateway interface) refer to a specification by which program can communicate with a web server.
- <http://www.icci.org/studies/ips.html> .
- 1. Protocol: http.
- 2. Host computer name: www.
- 3. Second-level domain name: icci.
- 4. Top-level domain name: org.
- 5. Directory name: studies.
- 6. File name: ips.html.

Several Top-level domain are common:

com: commercial enterprise. شركات

edu: educational institution. للمؤسسات التعليمية

gov: government entity. للمؤسسات الحكومية

mil: military entity. للمواقع العسكرية

net: network access provider. للمواقع ذات النشاط الخاص

org: usually nonprofit organizations منظمة رسمية غير حكومية

Classifying the Web Sites:

- There are several classify for the early Web sites in many terms, as the follow:
- Environment:
- The General Approach
- Classify in terms of Range of Complexity:

Environment:

- There are three main types of Web sites according to this classify: Internet, Intranet, and Extranet Web Sites.

A- Internet Web Sites:

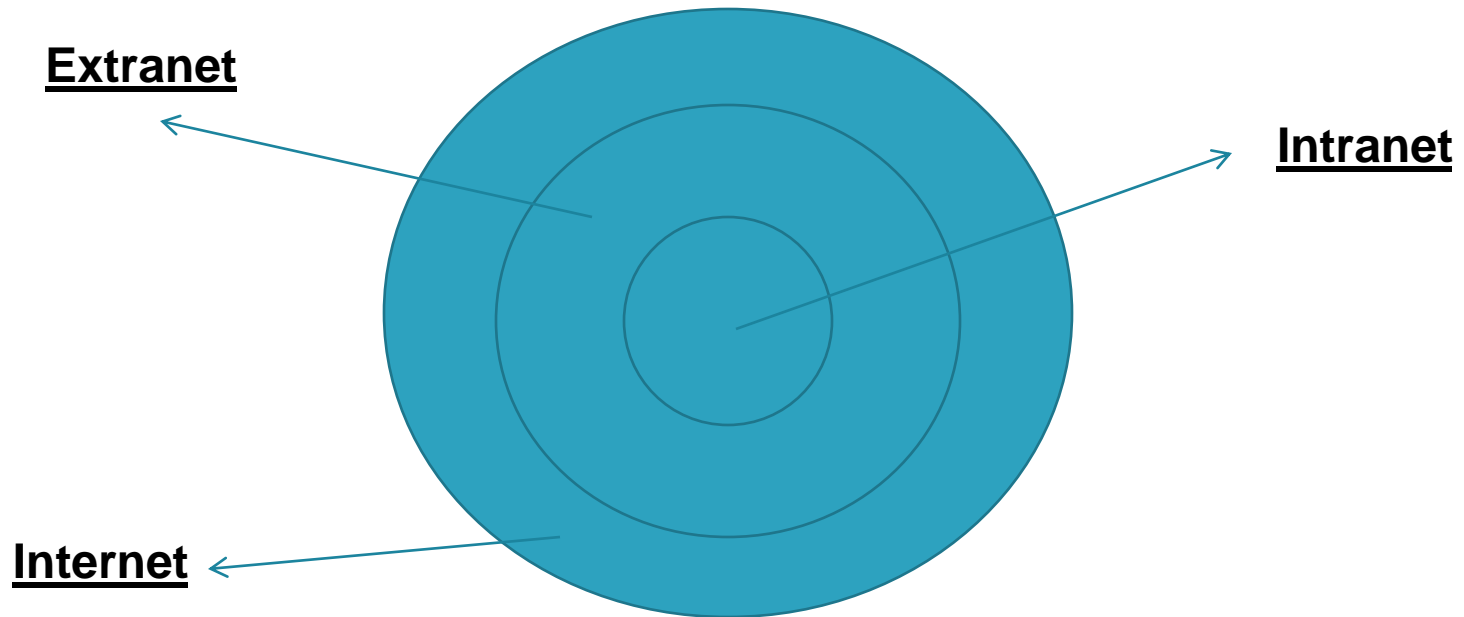
Internet Web Site is traditional Web sites that are intended for access by the general public.

B- Intranet Web Sites:

Intranet Web Site is intended only for internal (intra-organizational) use.

C- Extranet Web Sites:

Extranet Web Site is a combination of these. They are typically private and secured areas for the use of an organization and its designated partners.



The General Approach:

- There are two main types of Web sites according to this classify: Static and Dynamic Web sites.

A- Static Web Sites:

B- Dynamic Web Sites:

A- Static Web Sites:

- Static Web Site implies that the Web site will be a flat-file system of HTML files.
- all pages reside on the server and have fixed content that will be served “as is” to the user.

B- Dynamic Web Sites:

- Dynamically site requires that the content be stored in a database, not all sites require complete database functionality
- part of a site may be dynamic while others are static.

Classify in terms of Range of Complexity

There are five main types of Web sites according to this classify:

- Static Web Sites.
- Static with Form-Based Interactivity Web Sites.
- Static with Dynamic Data Access Web Sites.
- Dynamically Generated Web Sites.
- Web-Based Software Applications Web Sites.

Introduction to

HTML



Definitions

- W W W – World Wide Web.
- HTML – **HyperText Markup Language** – The Language of Web Pages on the World Wide Web.
HTML is a text formatting language.
- Browser – A software program which is used to show web pages.

- “Normal text” surrounded by bracketed *tags* that tell browsers how to display web pages
- Pages end with “.htm” or “.html”
- HTML Editor – A word processor that has been specialized to make the writing of HTML documents more effortless.

Tags

- Codes enclosed in brackets
- Usually paired

<TITLE>My Web Page</TITLE>

- ***Not*** case sensitive

<TITLE> = <title> = <TITLE>

Choosing Text Editor

- There are many different programs that you can use to create web documents.
- HTML Editors enable users to create documents quickly and easily by pushing a few buttons. Instead of entering all of the HTML codes by hand.
- These programs will generate the HTML Source Code for you.

Choosing Text Editor

- HTML Editors are excellent tools for experienced web developers; however; it is important that you learn and understand the HTML language so that you can edit code and fix “bugs” in your pages.
- For this Course, we will focus on using the standard Microsoft Windows text editors, NotePad. We may use also textpad.

Starting NotePad

NotePad is the standard text editor that comes with the microsoft windows operating system. To start NotePad in follow the steps bellow:

- Click on the “**Start**” button located on your Windows task bar.
- Click on “**Programs**” and then click on the directory menu labeled “**Accessories**”.
- Locate the shortcut “**NotePad**” and click the shortcut once.

Creating a Basic Starting Document

```
<HTML>
```

```
<HEAD>
```

```
  <TITLE>Al al-Bayt University</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
  This is what is displayed.
```

```
</BODY>
```

```
</HTML>
```

Previewing Your Work

- Once you have created your basic starting document and set your document properties it is a good idea to save your file.
- To save a file, in NotePad, follow these steps:
 1. Locate and click on the menu called “File”.
 2. Select the option under File Menu labeled “Save As”.
 3. In the “File Name” text box, type in the entire name of your file (including the extension name .html).

Creating a Basic Starting Document

- The HEAD of your document point to above window part. The TITLE of your document appears in the very top line of the user's browser. If the user chooses to "Bookmark" your page or save as a "Favorite"; it is the TITLE that is added to the list.
- The text in your TITLE should be as descriptive as possible because this is what many search engines, on the internet, use for indexing your site.

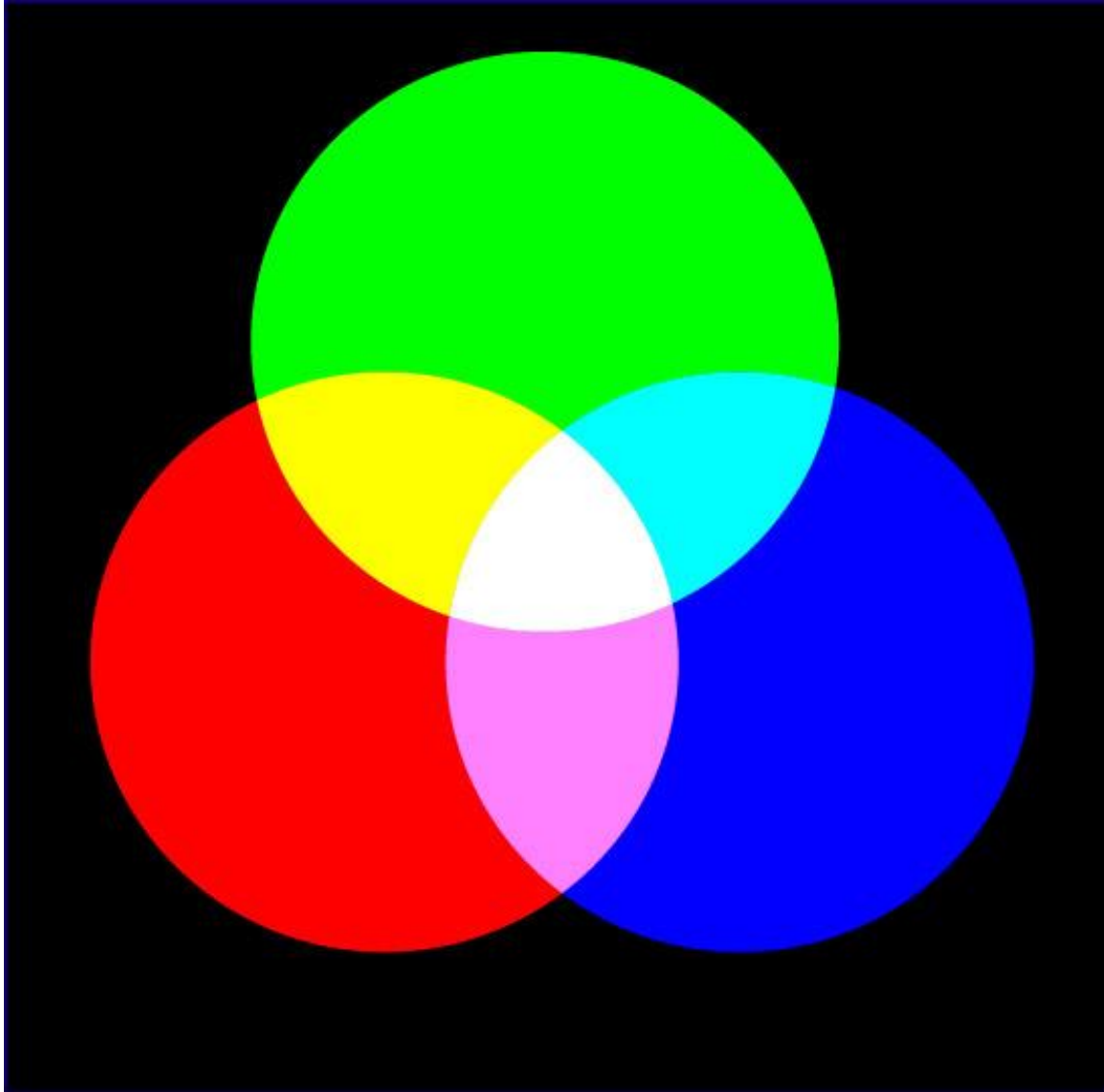
Setting Document Properties

- Document properties are controlled by attributes of the **BODY** element. For example, there are color settings for the background color of the page, the document's text and different states of links.

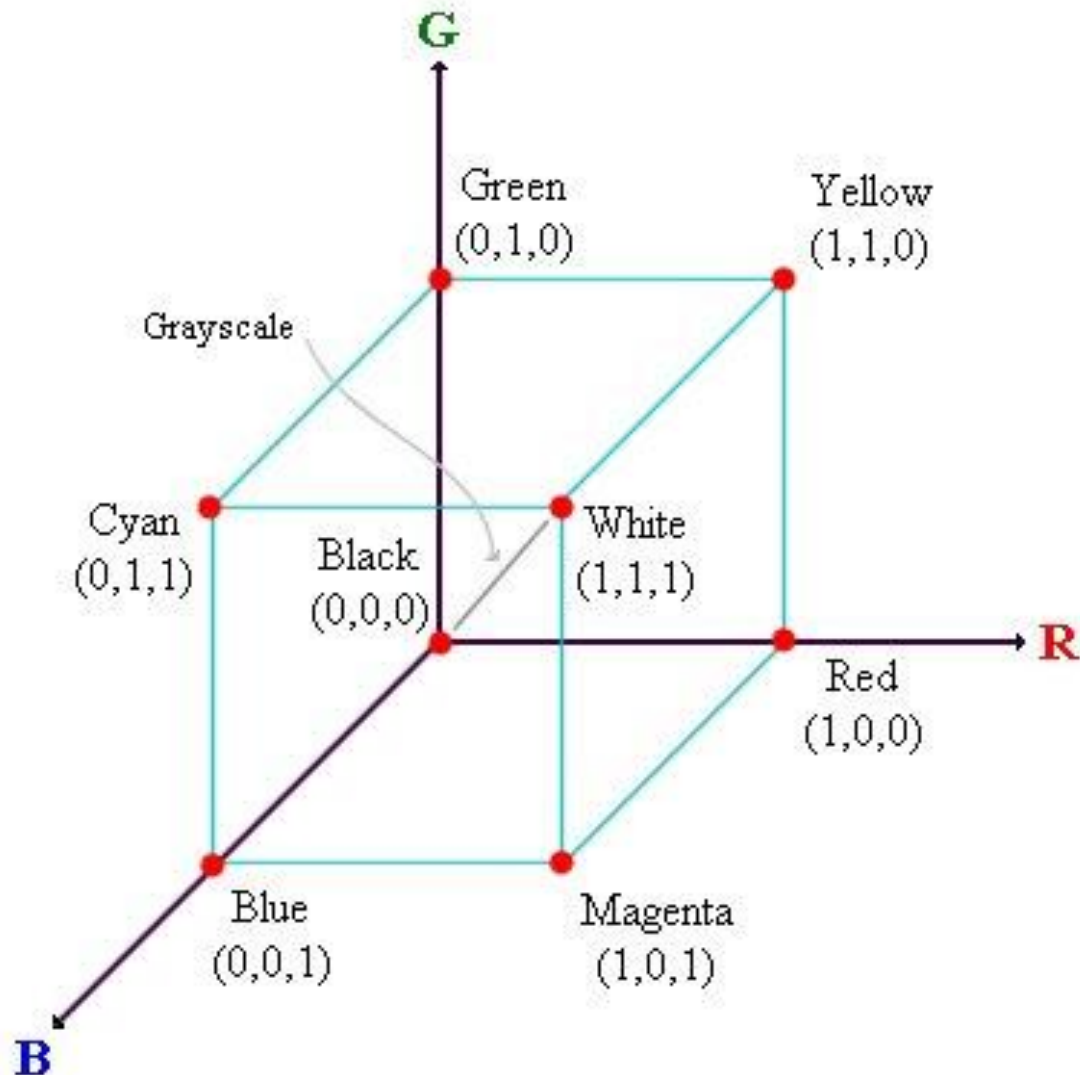
Color Codes

- Colors are set using “**RGB**” color codes, which are, represented as hexadecimal values. Each 2-digit section of the code represents the amount, in sequence, of **red**, **green** or **blue** that forms the color. For example, a **RGB** value with 00 as the first two digits has no red in the color.

Main Colours



RGB Colour Model



16 Basic Colors

Color Name	RGB Triplet	Hexadecimal	Color Name	RGB Triplet	Hexadecimal
Aqua	(0,255,255)	00FFFF	Navy	(0,0,128)	000080
Black	(0,0,0)	000000	Olive	(128,128,0)	808000
Blue	(0,0,255)	0000FF	Purple	(128,0,128)	800080
Fuchsia	(255,0,255)	FF00FF	Red	(255,0,0)	FF0000
Gray	(128,128,128)	808080	Silver	(192,192,192)	C0C0C0
Green	(0,128,0)	008000	Teal	(0,128,128)	008080
Lime	(0,255,0)	00FF00	White	(255,255,255)	FFFFFF
Maroon	(128,0,0)	800000	Yellow	(255,255,0)	FFFF00

Color Codes

1. WHITE
2. BLACK
3. RED
4. GREEN
5. BLUE
6. MAGENTA
7. CYAN
8. YELLOW
9. AQUAMARINE
10. BAKER'S CHOCOLATE
11. VIOLET
12. BRASS
13. COPPER
14. PINK
15. ORANGE

1. #FFFFFF
2. #000000
3. #FF0000
4. #00FF00
5. #0000FF
6. #FF00FF
7. #00FFFF
8. #FFFF00
9. #70DB93
10. #5C3317
11. #9F5F9F
12. #B5A642
13. #B87333
14. #FF6EC7
15. #FF7F00

The Body Element

- The **BODY** element of a web page is an important element in regards to the **page's appearance**. Here are the attributes of the **BODY** tag to control all the levels:

TEXT="#RRGGBB" to change the color of **all the text** on the page (**full page text color.**)

- This element contains information about the page's background color, the background image, as well as the text and link colors.

Background Color

- It is very common to see web pages with their background color set to white or some other colors.
- To set your document's background color, you need to edit the <BODY> element by adding the BGCOLOR attribute. The following example will display a document with a white background color:

```
<BODY BGCOLOR="#FFFFFF"></BODY>
```

TEXT Color

- The TEXT attribute is used to control the color of all the normal text in the document. The default color for text is black. The TEXT attribute would be added as follows:

```
<BODY BGCOLOR="#FFFFFF"  
TEXT="#FF0000"></BODY>
```

In this example the document's page color is white and the text would be red.

LINK, VLINK, and ALINK

These attributes control the colors of the different link states:

1. LINK – initial appearance – default = Blue.
2. VLINK – visited link – default = Purple.
3. ALINK – active link being clicked – default = Yellow.

The Format for setting these attributes is:

```
<BODY BGCOLOR="#FFFFFF" TEXT="#FF0000"  
  LINK="#0000FF"  
  VLINK="#FF00FF"  
  ALINK="FFFF00"> </BODY>
```

Using Image Background

- The BODY element also gives you ability of setting an image as the document's background.
- An example of a background image's HTML code is as follows:

```
<BODY BACKGROUND="hi.gif"  
  BGCOLOR="#FFFFFF"></BODY>
```

Headings, `<Hx>` `</Hx>`

- Inside the **BODY** element, heading elements **H1** through **H6** are generally used for major divisions of the document. Headings are permitted to appear in any order, but you will obtain the best results when your documents are displayed in a browser if you follow these guidelines:
 1. **H1**: should be used as the highest level of heading, **H2** as the next highest, and so forth.
 2. You should not skip heading levels: e.g., an **H3** should not appear after an **H1**, unless there is an **H2** between them.

Headings, <Hx> </Hx>

```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<H2> Heading 2 </H2>
<H3> Heading 3 </H3>
<H4> Heading 4 </H4>
<H5> Heading 5 </H5>
<H6> Heading 6 </H6>
</BODY>
</HTML>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Paragraphs, `<P>` `</P>`

- Paragraphs allow you to add text to a document in such a way that it will automatically adjust the end of line to suite the window size of the browser in which it is being displayed. Each line of text will stretch the entire length of the window.

Paragraphs, <P> </P>

```
<HTML><HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY></H1> Heading 1 </H1>
<P> Paragraph 1, ....</P>
<H2> Heading 2 </H2>
<P> Paragraph 2, ....</P>
<H3> Heading 3 </H3>
<P> Paragraph 3, ....</P>
<H4> Heading 4 </H4>
<P> Paragraph 4, ....</P>
<H5> Heading 5 </H5>
<P> Paragraph 5, ....</P>
<H6> Heading 6</H6>
<P> Paragraph 6, ....</P>
</BODY></HTML>
```

Heading 1

Paragraph 1,....

Heading 2

Paragraph 2,....

Heading 3

Paragraph 3,....

Heading 4

Paragraph 4,....

Heading 5

Paragraph 5,....

Heading 6

Paragraph 6,....

Break,

- Line breaks allow you to decide where the text will break on a line or continue to the end of the window.
- A
 is an empty Element, meaning that it may contain attributes but it does not contain content.
- The
 element does not have a closing tag.

Break,


```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<P>Paragraph 1, <BR>
Line 2 <BR> Line 3 <BR>....
</P>
</BODY>
</HTML>
```

Heading 1

Paragraph 1,.....

Line 2

Line 3

.....

Horizontal Rule, <HR>

- The <HR> element causes the browser to display a horizontal line (rule) in your document.
- <HR> does not use a closing tag, </HR>.

Horizontal Rule, <HR>

```
<HTML>
<HEAD>
<TITLE> Example Page</TITLE>
</HEAD>
<BODY>
<H1> Heading 1 </H1>
<P>Paragraph 1, <BR>
Line 2 <BR>
<HR>Line 3 <BR>
</P>
</BODY>
</HTML>
```

Heading 1

Paragraph 1,.....

Line 2

Line 3

Bold, Italic and other Character Formatting Elements

- **** Two sizes bigger****
- The size attribute can be set as an absolute value from 1 to 7 or as a relative value using the "+" or "-" sign. Normal text size is 3 (from -2 to +4).
- ** Bold **
- **<I> *Italic* </I>**
- **<U> Underline </U>**
- Color = "#RRGGBB" The COLOR attribute of the FONT element. E.g., **this text has color**

Bold, Italic and other Character Formatting Elements

<P> One
Size Larger - Normal
–

 One Size
Smaller

 Bold - <I> italics</I> -
<U> Underlined </U> -

Colored

One Size Larger - Normal – One
Size Smaller

Bold - *italics* - Underlined -
Colored

Alignment

- Some elements have attributes for alignment (ALIGN) e.g. **Headings, Paragraphs and Horizontal Rules.**
- The Three alignment values are : LEFT, RIGHT, CENTER.
- **<CENTER></CENTER>** Will center elements.

Special Characters & Symbols

Special Character	Entity Name	Special Character	Entity Name
Ampersand	&amp; &	Greater-than sign	&gt; >
Asterisk	&lowast; **	Less-than sign	&lt; <
Cent sign	&cent; ¢	Non-breaking space	&nbsp; ;
Copyright	&copy; ©	Quotation mark	&quot; "
Fraction one qtr	&frac14; $\frac{1}{4}$	Registration mark	&reg; ®
Fraction one half	&frac12; $\frac{1}{2}$	Trademark sign	&trade; ™

Lists

In this chapter you will learn how to create a variety of lists.

Objectives

Upon completing this section, you should be able to

1. Create an unordered list.
2. Create an ordered list.
3. Create a defined list.
4. Nest Lists.

List Elements

- HTML supplies several list elements. Most list elements are composed of one or more (List Item) elements.
- UL : Unordered List. Items in this list start with a list mark such as a bullet. Browsers will usually change the list mark in nested lists.

 List item ...

 List item ...

- List item ...
- List item ...

List Elements

- You have the choice of three bullet types: **disc(default)**, **circle**, **square**.
- These are controlled in Netscape Navigator by the “TYPE” attribute for the element.

```
<UL TYPE="square">
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
</UL>
```

- List item ...
- List item ...
- List item ...

List Elements

- OL: Ordered List. Items in this list are numbered automatically by the browser.

 List item ...

 List item ...

 List item ...

1. **List item ...**

2. **List item ...**

3. **List item**

- You have the choice of setting the TYPE Attribute to one of five numbering styles.

List Elements

TYPE	Numbering Styles	
1	Arabic numbers	1,2,3,
a	Lower alpha	a, b, c,
A	Upper alpha	A, B, C,
i	Lower roman	i, ii, iii,
I	Upper roman	I, II, III,

List Elements

- You can specify a starting number for an ordered list.

```
<OL TYPE =“i”>
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
</OL>
```

```
<P> text ....</P>
```

```
<OL TYPE=“i” START=“3”>
```

```
<LI> List item ...</LI>
```

```
</OL>
```

List Elements

i. List item ...

ii. List item ...

Text

iii. List item ...

List Elements

- **DL: Definition List.** This kind of list is different from the others. Each item in a DL consists of one or more **Definition Terms (DT elements)**, followed by one or more **Definition Description (DD elements)**.

```
<DL>
```

```
<DT> HTML </DT>
```

```
<DD> Hyper Text Markup Language </DD>
```

```
<DT> DOG </DT>
```

```
<DD> A human's best friend!</DD>
```

```
</DL>
```

HTML

Hyper Text Markup Language

DOG

A human's best friend!

Nesting Lists

- You can nest lists by inserting a UL, OL, etc., inside a list item (LI).

Example

```
<UL TYPE = "square">
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...
```

```
<OL TYPE="i" START="3">
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
<LI> List item ...</LI>
```

```
</OL>
```

```
</LI>
```

```
<LI> List item ...</LI>
```

```
</UL>
```

- List item ...
- List item ...
 - iii. List item ...
 - iv. List item ...
 - v. List item ...
 - vi. List item ...
 - vii. List item ...
- List item ...

What will be the output?

```
<H1 ALIGN="CENTER">SAFETY TIPS FOR CANOEISTS</H1>
<OL TYPE="a" START="2">
<LI>Be able to swim </LI>
<LI>Wear a life jacket at all times </LI>
<LI>Don't stand up or move around. If canoe tips,
  <UL>
    <LI>Hang on to the canoe </LI>
    <LI>Use the canoe for support and </LI>
    <LI>Swim to shore
  </UL> </LI>
<LI>Don't overexert yourself </LI>
<LI>Use a bow light at night </LI>
</OL>
```

Images

In this chapter you will learn about images and how to place images in your pages.

Objectives

Upon completing this section, you should be able to

1. Add images to your pages.

Images

- **** This element defines a graphic image on the page.
- **Image File (SRC:source):** This value will be a URL (location of the image) E.g. <http://www.domain.com/dir/file.ext> or /dir/file.txt.
- **Alternate Text (ALT):** This is a text field that describes an image or acts as a label. It is displayed when they position the cursor over a graphic image.
- **Alignment (ALIGN):** This allows you to align the image on your page.

Images

- **Width (WIDTH):** is the width of the image in pixels.
- **Height (HEIGHT):** is the height of the image in pixels.
- **Border (BORDER):** is for a border around the image, specified in pixels.

Some Examples on images

- 1) ``
- 2) ``
- 3) ``
- 4) ``
- 5) `< IMG SRC =" jordan.gif" align="left">`
blast blast blast blast blast

Anchors, URLs and Image Maps

In this chapter you will learn about Uniform Resource Locator, and how to add them as Anchor or Links inside your web pages.

Objectives

Upon completing this section, you should be able to

1. Insert links into documents.
2. Define Link Types.
3. Define URL.
4. List some commonly used URLs.
5. Plan an Image Map.

HOW TO MAKE A LINK

1) The tags used to produce links are the `<A>` and ``. The `<A>` tells where the link should start and the `` indicates where the link ends. Everything between these two will work as a link.

2) The example below shows how to make the word **Here** work as a link to yahoo.

Click `here` to go to yahoo.

More on LINKs

```
<body LINK="#C0C0C0" VLINK="#808080"  
ALINK="#FF0000">
```

- **LINK** - standard link - to a page the visitor hasn't been to yet. (standard color is blue - #0000FF).
- **VLINK** - visited link - to a page the visitor has been to before. (standard color is purple - #800080).
- **ALINK** - active link - the color of the link when the mouse is on it. (standard color is red - #FF0000).

If the programmer what to change the color

- Click `here` to go to yahoo.

E-Mail (Electronic Mail)

E.g. <mailto:kmf@yahoo.com>

- The type of service is identified as the mail client program. This type of link will launch the users mail client.
- The recipient of the message is kmf@yahoo.com

Send me
More Information

Image Maps

- Image maps are images, usually in **gif** format that have been divided into regions; clicking in a region of the image cause the web surfer to be connected to a new URL. Image maps are graphical form of creating links between pages.
- There are two type of image maps:

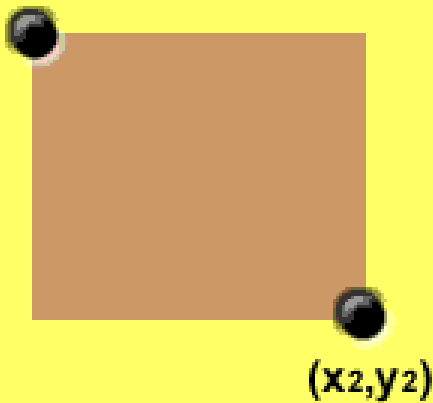
Client side and server side

Both types of image maps involve a listing of co-ordinates that define the mapping regions and which URLs those coordinates are associated with. This is known as the map file.

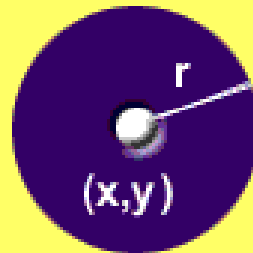
Area Shapes Used

Finding the coordinates....

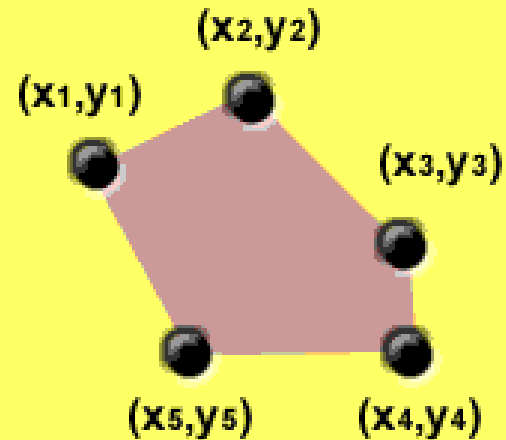
(x_1, y_1)



Rectangle



Circle



Polygon

Client-Side Image Maps

- Client-side image maps (USEMAP) use a map file that is part of the HTML document (in an element called MAP), and is linked to the image by the Web browser.

```
<IMG SRC="note.GIF" Width=200 Height=200  
border="5" USEMAP="#map1">
```

```
<MAP NAME="map1">
```

```
<AREA SHAPE="RECT" COORDS="0,0,90,90"  
HREF="hi.html" ALT="see me...">
```

```
<AREA SHAPE="RECT" COORDS="100,100,160,160"  
HREF="divPara.html" ALT="see him..." >
```

```
<AREA SHAPE="CIRCLE" COORDS="150,50,20"  
HREF="house.html" ALT="see it..." >
```

```
</MAP>
```

We can use Poly as well as Rect.....

Shapes, Coords

- Types of Shapes
 - Rect → used for squares and ordered shapes.
 - Circle → used for circles.
 - Poly → used for unordered shapes.
- Number of coordinations for each shape:
 - Rect → 4 numbers for two corners
 - Circle → 3 numbers for the center & R
 - Poly → depends on the number of corners of the shape(2 numbers for each corner)

Tables

In this chapter you will learn that tables have many uses in HTML.

Objectives:

Upon completing this section, you should be able to:

1. Insert a table.
2. Explain a table's attributes.
3. Edit a table.
4. Add a table header.

Tables

- The `<TABLE></TABLE>` element has four sub-elements:
 1. Table Row `<TR></TR>`.
 2. Table Header `<TH></TH>`.
 3. Table Data `<TD></TD>`.
 4. Caption `<CAPTION></CAPTION>`.
- The table row elements usually contain table header elements or table data elements.

Tables

```
<table border="1">  
<tr>  
<th> Column 1 header </th>  
<th> Column 2 header </th>  
</tr>  
<tr>  
<td> Row1, Col1 </td>  
<td> Row1, Col2 </td>  
</tr>  
<tr>  
<td> Row2, Col1 </td>  
<td> Row2, Col2 </td>  
</tr>  
</table>
```

Tables

Column 1 Header	Column 2 Header
Row1, Col1	Row1, Col2
Row2, Col1	Row2, Col2

Tables Attributes

- **BGColor**: Some browsers support background colors in a table.
- **Width**: you can specify the table width as an absolute number of pixels or a percentage of the document width. You can set the width for the table cells as well.
- **Border**: You can choose a numerical value for the border width, which specifies the border in pixels.
- **CellSpacing**: Cell Spacing represents the space between cells and is specified in pixels.

Table Attributes

- **CellPadding**: Cell Padding is the space between the cell border and the cell contents and is specified in pixels.
- **Align**: tables can have left, right, or center alignment.
- **Background**: Background Image, will be titled in IE3.0 and above.
- **BorderColor, BorderColorDark**.

Table Caption

- A table caption allows you to specify a line of text that will appear centered above or below the table.

```
<TABLE BORDER=1 CELLPADDING=2>
```

```
<CAPTION ALIGN="BOTTOM"> Label For My Table  
</CAPTION>
```

- The Caption element has one attribute ALIGN that can be either TOP (Above the table) or BOTTOM (below the table).

Table Header

- Table Data cells are represented by the TD element. Cells can also be TH (Table Header) elements which results in the contents of the table header cells appearing **centered and in bold text**.

Table Data and Table Header Attributes

- **Colspan:** Specifies how many cell columns of the table this cell should span.
- **Rowspan:** Specifies how many cell rows of the table this cell should span.
- **Align:** cell data can have left, right, or center alignment.
- **Valign:** cell data can have top, middle, or bottom alignment.
- **Width:** you can specify the width as an absolute number of pixels or a percentage of the document width.
- **Height:** You can specify the height as an absolute number of pixels or a percentage of the document height.

Basic Table Code

```
<TABLE BORDER=1 width=50%>
<CAPTION> <h1>Spare Parts </h1> </Caption>
<TR><TH>Stock Number</TH><TH>Description</TH><TH>List
Price</TH></TR>
<TR><TD bgcolor=red>3476-AB</TD><TD>76mm
Socket</TD><TD>45.00</TD></TR>
<TR><TD >3478-AB</TD><TD><font color=blue>78mm Socket</font>
</TD><TD>47.50</TD></TR>
<TR><TD>3480-AB</TD><TD>80mm Socket</TD><TD>50.00</TD></TR>
</TABLE>
```

Spare Parts

Stock Number	Description	List Price
3476-AB	76mm Socket	45.00
3478-AB	78mm Socket	47.50
3480-AB	80mm Socket	50.00

Table Data and Table Header Attributes

```
<Table border=1 cellpadding =2>  
<tr> <th> Column 1 Header</th> <th>  
Column 2 Header</th> </tr>  
<tr> <td colspan=2> Row 1 Col 1</td> </tr>  
<tr> <td rowspan=2>Row 2 Col 1</td>  
<td> Row 2 Col2</td> </tr>  
<tr> <td> Row 3 Col2</td> </tr>  
</table>
```

Table Data and Table Header Attributes

Column 1 Header	Column 2 Header
Row 1 Col 1	
Row 2 Col 1	Row 2 Col 2
	Row 3 Col 2

Special Things to Note

- **TH, TD and TR should always have end tags.**
Although the end tags are formally optional, many browsers will mess up the formatting of the table if you omit the end tags. In particular, you should *always* use end tags if you have a TABLE within a TABLE -- in this situation, the table parser gets hopelessly confused if you don't close your TH, TD and TR elements.
- **A default TABLE has no borders**
By default, tables are drawn without border lines. You need the BORDER attribute to draw the lines.
- **By default, a table is flush with the left margin**
TABLEs are plopped over on the left margin. If you want centered tables, You can either: place the table inside a DIV element with attribute ALIGN="center".
Most current browsers also supports table alignment, using the ALIGN attribute. Allowed values are "left", "right", or "center", for example: <TABLE ALIGN="left">. The values "left" and "right" float the table to the left or right of the page, with text flow allowed around the table. This is entirely equivalent to IMG alignment

What will be the output?

```
<TABLE BORDER width="750">
```

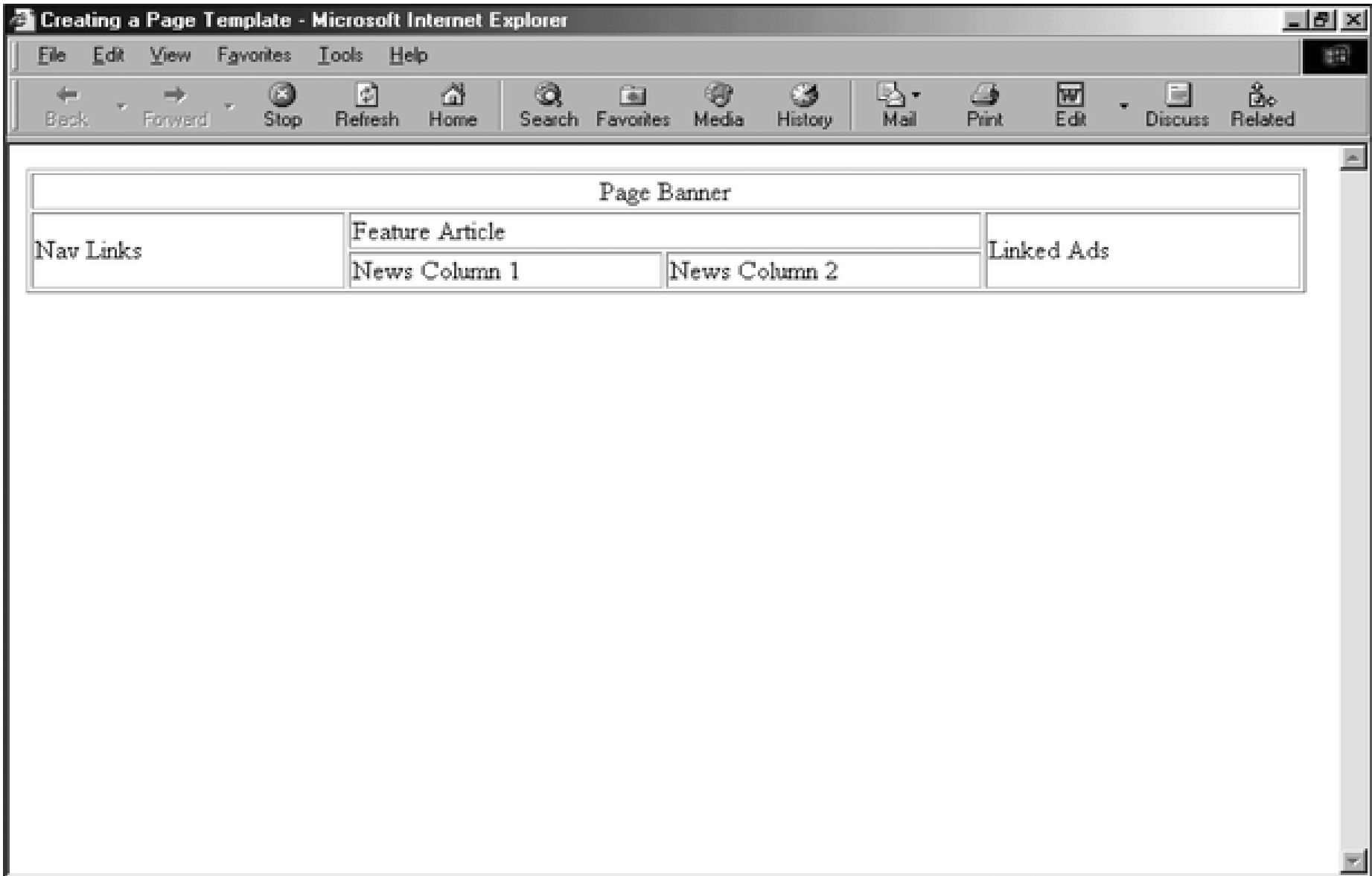
```
<TR> <TD colspan="4" align="center">Page  
Banner</TD></TR>
```

```
<TR> <TD rowspan="2" width="25%">Nav  
Links</TD><TD colspan="2">Feature  
Article</TD> <TD rowspan="2"  
width="25%">Linked Ads</TD></TR>
```

```
<TR><TD width="25%">News Column 1 </TD>  
<TD width="25%"><News Column 2 </TD></TR>
```

```
</TABLE>
```

The Output



Frames

- Frames are a relatively new addition to the HTML standard. First introduced in Netscape Navigator 2.0.

Objectives:

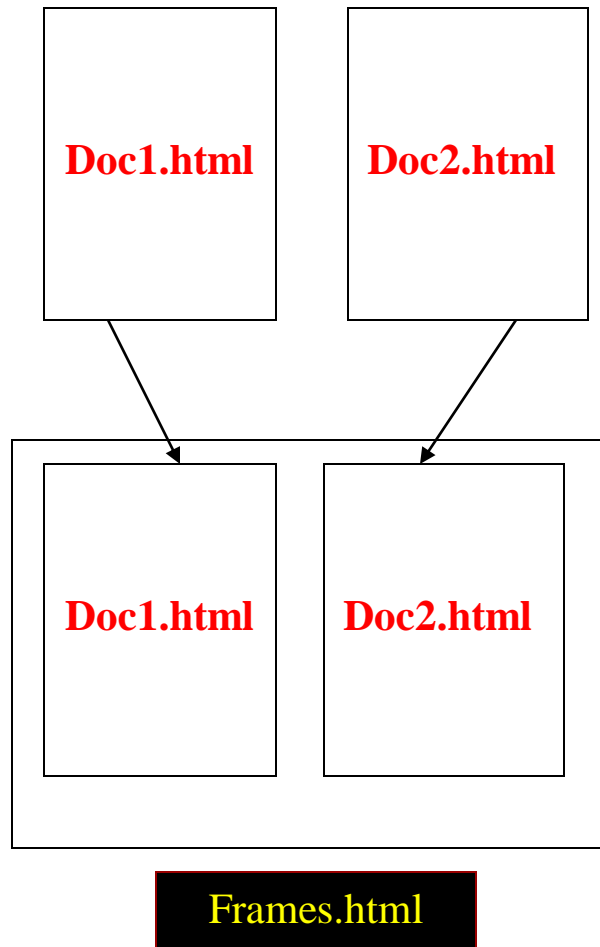
Upon completing this section, you should be able to:

- Create a Frame based page.
- Work with the Frameset, Frame, and Noframes elements.
- Use the attributes of the Frames elements to control the display.
- Set Targets appropriately.

Frames

- A framed page is actually made up of multiple HTML pages. There is one HTML document that describes how to break up the single browser window into multiple windowpanes. Each windowpane is filled with an HTML document.
- For Example to make a framed page with a windowpane on the left and one on the right requires three HTML pages. ***Doc1.html*** and ***Doc2.html*** are the pages that contain content. ***Frames.html*** is the page that describes the division of the single browser window into two windowpanes.

Frames



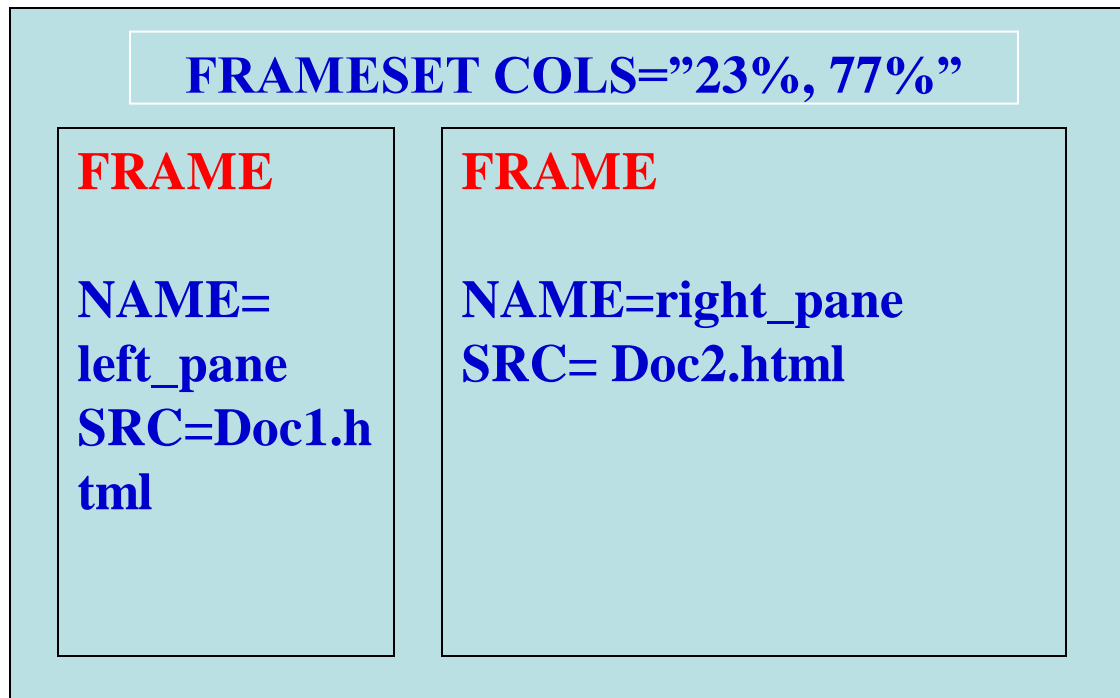
Frame Page Architecture

- A **<FRAMESET>** element is placed in the html document before the **<BODY>** element. The **<FRAMESET>** describes the amount of screen real estate given to each windowpane by dividing the screen into **ROWS** or **COLS**.
- The **<FRAMESET>** will then contain **<FRAME>** elements, **one per division** of the browser window.
- Note: Because there is no **BODY** container, FRAMESET pages can't have **background images** and **background colors** associated with them.

Frame Page Architecture

```
<HTML>  
<HEAD>  
<TITLE> Framed Page </TITLE>  
<FRAMESET COLS="23%,77%">  
<FRAME SRC="Doc1.html">  
<FRAME SRC="Doc2.html">  
</FRAMESET >  
</HEAD>  
  
</HTML>
```

The Diagram below is a graphical view of the document described above



<FRAMESET> Container

<FRAMESET> : The FRAMESET element creates divisions in the browser window in a single direction. This allows you to define divisions as either rows or columns.

- **ROWS** : Determines the size and number of rectangular rows within a <FRAMESET>. They are set from top of the display area to the bottom.

Possible values are:

- Absolute pixel units, I.e. “360,120”.
- A percentage of screen height, e.g. “75%,25%”.
- Proportional values using the asterisk (*). This is often combined with a value in pixels , e.g. “360,*” .
- <Frameset cols=“200,20%,*,2*”>

Creating a Frames Page

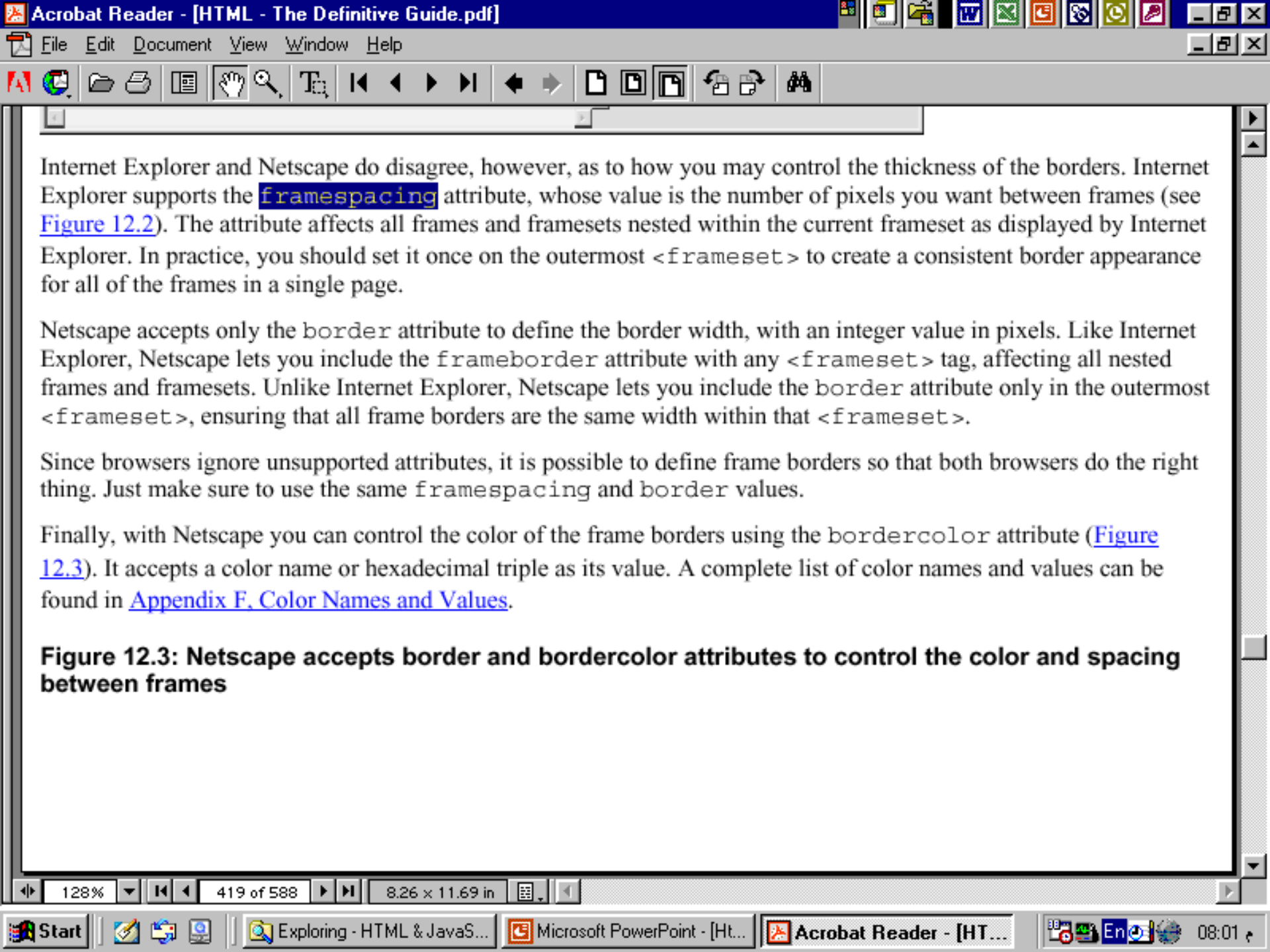
- **COLS**: Determines the size and number of rectangular columns within a <FRAMESET>. They are set from **left** to **right** of the display area.

Possible values are:

- Absolute pixel units, I.e. “480,160”.
- A percentage of screen width, e.g. “75%,25%”.
- Proportional values using the asterisk (*). This is often combined with a value in pixels , e.g. “480,*”.

Creating a Frames Page

- **FRAMEBORDER** : Possible values **0, 1, YES, NO**. A setting of zero will create a borderless frame.
- **FRAMESPACING**: This attribute is specified in **pixels**. If you go to borderless frames you will need to set this value to zero as well, or you will have a gap between your frames where the border used to be.
- **BORDER(thickness of the Frame)**: This attribute specified in pixels. A setting of zero will create a borderless frame. Default value is 5.
- **BORDERCOLOR**: This attribute is allows you choose a color for your border. This attribute is rarely used.



Internet Explorer and Netscape do disagree, however, as to how you may control the thickness of the borders. Internet Explorer supports the `framespacing` attribute, whose value is the number of pixels you want between frames (see [Figure 12.2](#)). The attribute affects all frames and framesets nested within the current frameset as displayed by Internet Explorer. In practice, you should set it once on the outermost `<frameset>` to create a consistent border appearance for all of the frames in a single page.

Netscape accepts only the `border` attribute to define the border width, with an integer value in pixels. Like Internet Explorer, Netscape lets you include the `frameborder` attribute with any `<frameset>` tag, affecting all nested frames and framesets. Unlike Internet Explorer, Netscape lets you include the `border` attribute only in the outermost `<frameset>`, ensuring that all frame borders are the same width within that `<frameset>`.

Since browsers ignore unsupported attributes, it is possible to define frame borders so that both browsers do the right thing. Just make sure to use the same `framespacing` and `border` values.

Finally, with Netscape you can control the color of the frame borders using the `bordercolor` attribute ([Figure 12.3](#)). It accepts a color name or hexadecimal triple as its value. A complete list of color names and values can be found in [Appendix F, Color Names and Values](#).

Figure 12.3: Netscape accepts `border` and `bordercolor` attributes to control the color and spacing between frames

<FRAME>

<FRAME>: This element defines a single frame within a frameset. There will be a FRAME element for each division created by the FRAMESET element. This tag has the following attributes:

- **SRC**: Required, as it provides the URL for the page that will be displayed in the frame.
- **NAME**: Required for frames that will allow targeting by other HTML documents. Works in conjunction with the target attribute of the <A>, <AREA>, <BASE>, and <FORM> tags.

<FRAME>

- **MARGINWIDTH**: Optional attribute stated in pixels. Determines horizontal space between the <FRAME> contents and the frame's borders.
- **MARGINHEIGHT**: Optional attribute stated in pixels. Determines vertical space between the <FRAME> contents and the frame's borders.
- **SCROLLING**: Displays a scroll bar(s) in the frame. Possible values are:
 1. **Yes** – always display scroll bar(s).
 2. **No** – never display scroll bar(s).
 3. **Auto** – browser will decide based on frame contents.

By default: scrolling is auto.

<FRAME>

- **NORESIZE**: Optional – prevents viewers from resizing the frame. By default the user can stretch or shrink the frame's display by selecting the frame's border and moving it up, down, left, or right.

<NOFRAMES>

- **<NOFRAMES>**: Frame – capable browsers ignore all HTML within this tag including the contents of the BODY element. This element does not have any attributes.

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> Framed Page </TITLE>
```

```
</HEAD>
```

<NOFRAMES>

```
<FRAMESET COLS="23%,77%">
```

```
<FRAME SRC="" NAME="left_pane">
```

```
<FRAME SRC="" NAME="right_pane">
```

```
<NOFRAMES>
```

```
<P> This is a Framed Page. Upgrade your  
browser to support frames.</P>
```

```
</NOFRAMES></FRAMESET>
```

Compound FRAMESET Divisions

- In this case a second **FRAMESET** element will be inserted in the place of the **FRAME** element that would describe the second row.
- The second **FRAMESET** element will divide the remaining screen real estate into **2** columns.
- This nested **FRAMESET** will then be followed by **2 FRAME** elements to describe each of the subsequent frame divisions created.

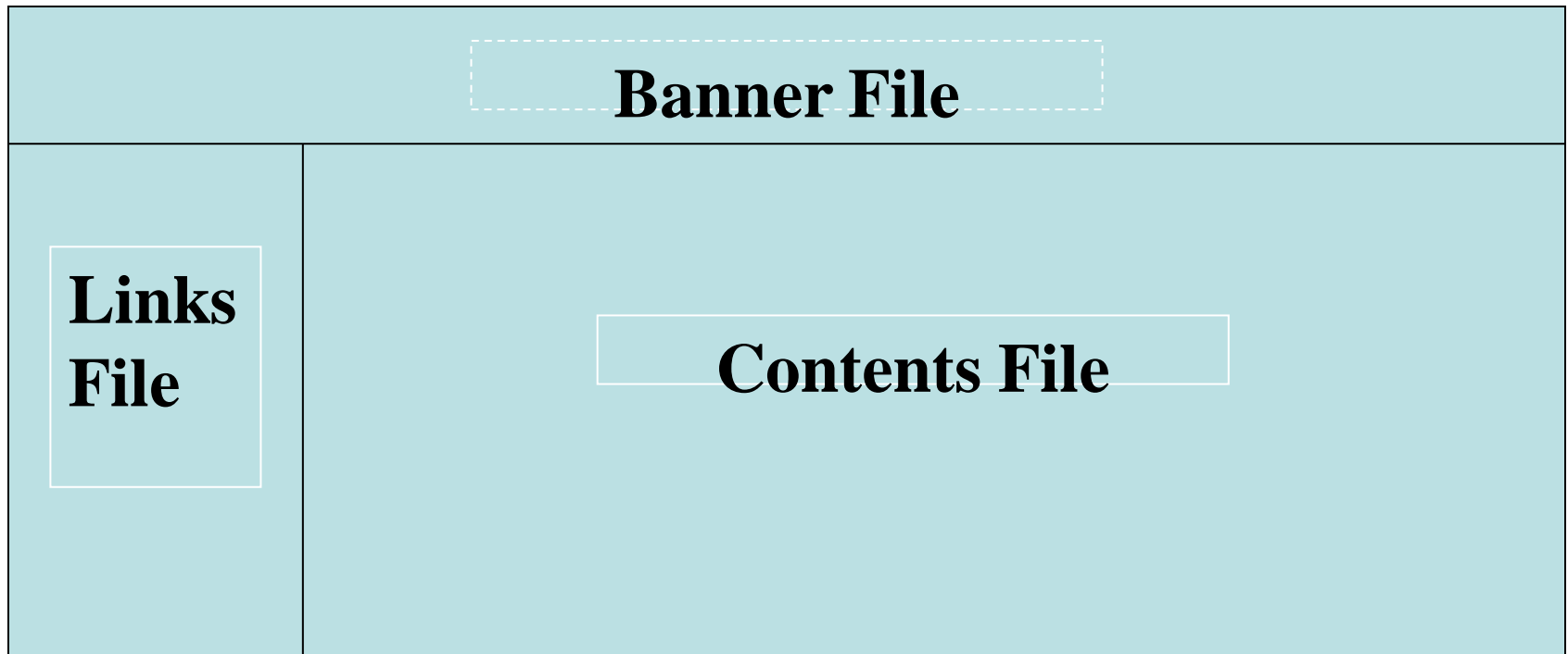
Compound FRAMESET Divisions

```
<html>
<head>
<title> Compound Frames Page</title>
</head>
<frameset rows="120,*">
<frame src="banner_file.html"
  name"banner">
<frameset cols="120,*">
<frame src="links_file.html"
  name="links">
<frame src="content_file.html"
  name="content">
```

```
<noframes>
<p>
Default
  message
</p>
</noframes>
</frameset>
</frameset>
</head>
```

Compound FRAMESET Divisions

You may want to create a frames design with a combination of rows and columns.



Compound FRAMESET Divisions Example

<HEAD>

<FRAMESET ROWS="25%,50%,25%"

<FRAME SRC="">

<FRAMESET COLS="25%,*">

<FRAME SRC="">

<FRAME SRC="">

</FRAMESET>

<FRAME SRC="">

</FRAMESET>

</HEAD>

Output



<< Links انتشغال < >

C:\Documents and Settings\Khaled\المكتب\سطح المكتب\HTMLcourse\Web_Page_Design\HTMLExamples\tp026561.html

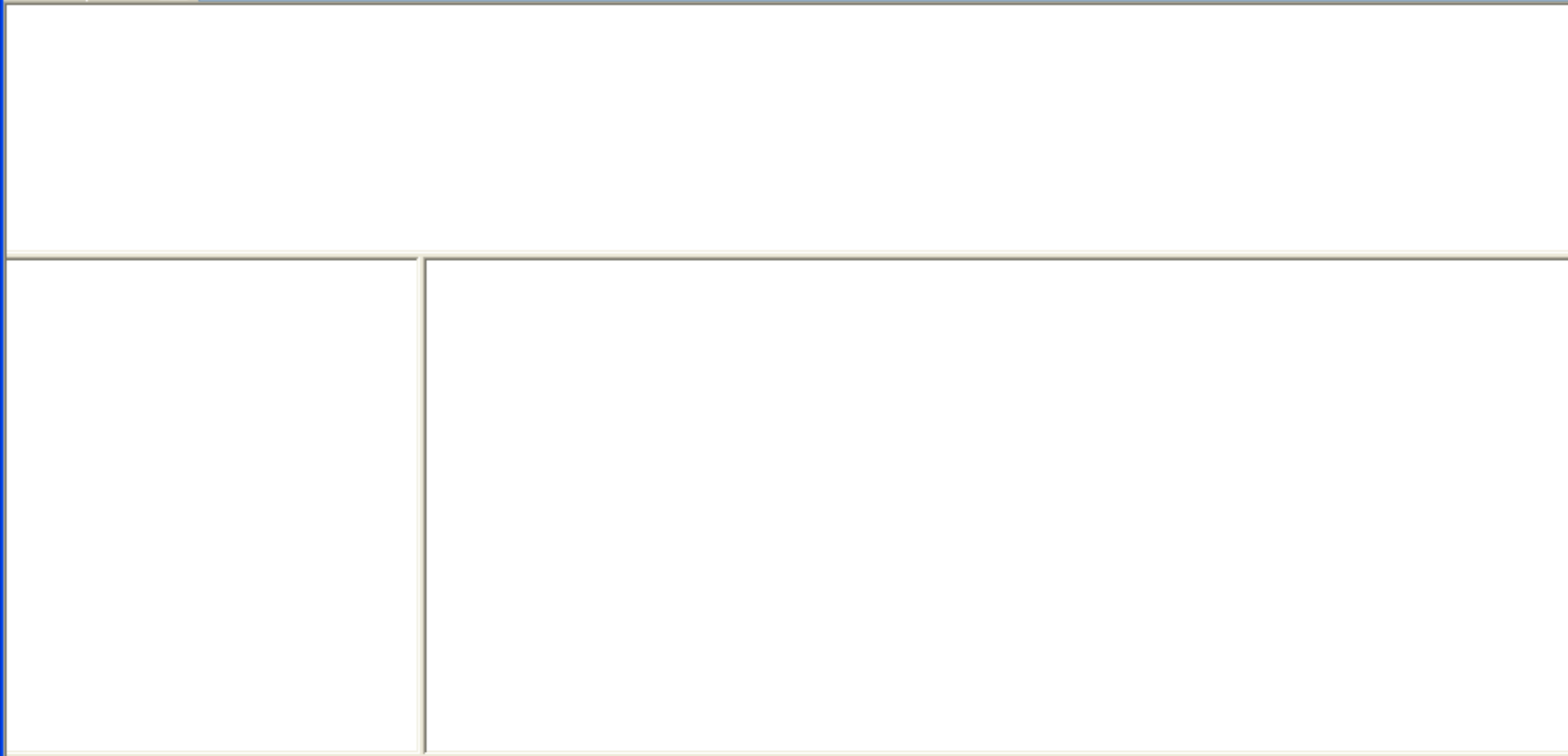




Figure 5-14: Frames created with `<FRAMESET ROWS="50%, 50%">`

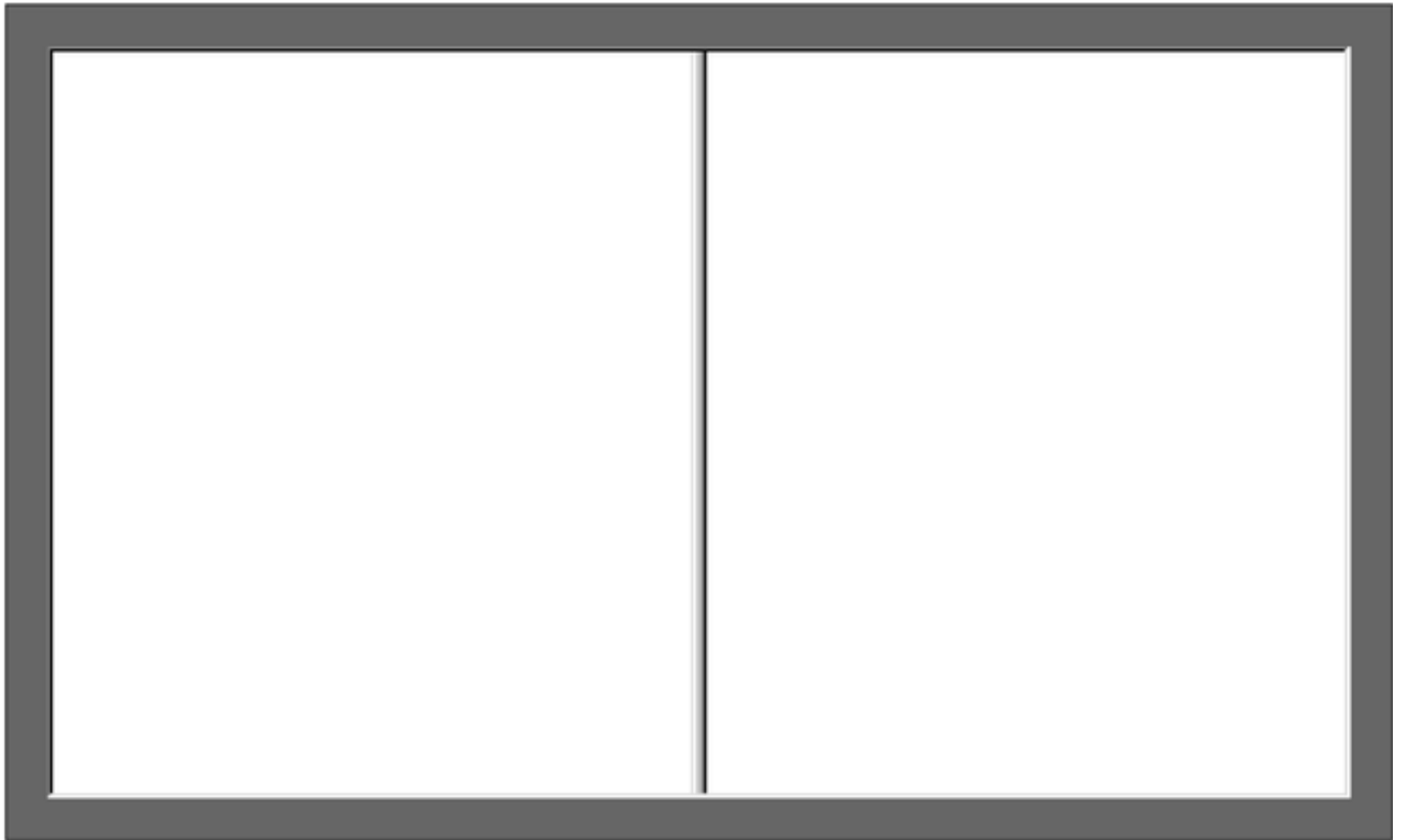


Figure 5-15: Frames created with `<FRAMESET COLS="50%, 50%">`

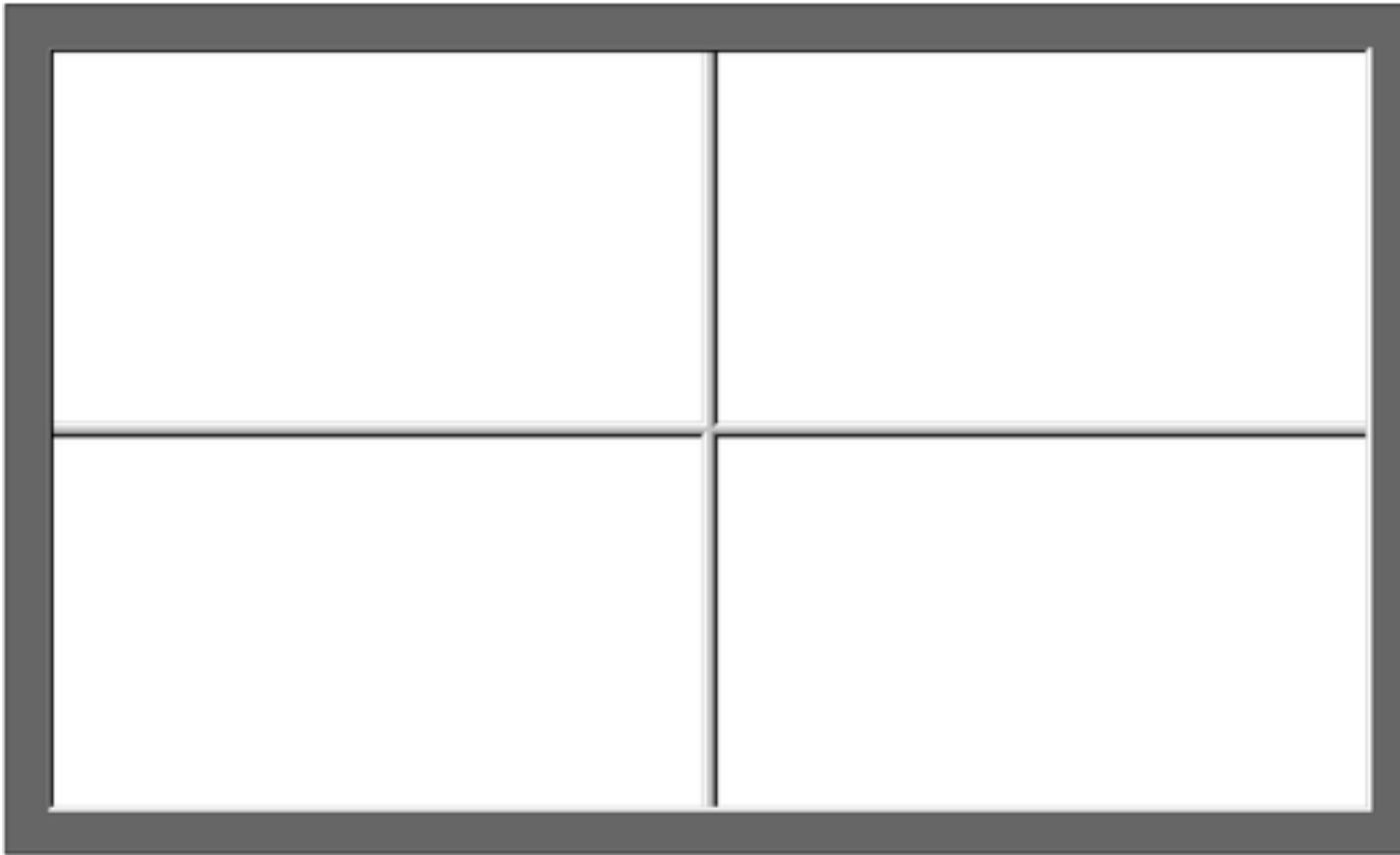


Figure 5-13: Frames created with `<FRAMESET ROWS="50%, 50%" COLS="50%, 50%">`

Frame Formatting

- Example:

```
<frameset rows="20%, *, 20%">
```

```
    <frame src="header.html" noresize  
    scrolling=no>
```

```
    <frame src="body.html">
```

```
    <frame src="navigationbar.html"  
    noresize scrolling=no>
```

```
</frameset>
```


What do the following mean?

- 1) <FRAMESET COLS="2*, 3*, 5*">
- 2) <FRAMESET COLS="150, 20%, *, 3*">

So what are the space-allocation priorities?

Absolute pixel values are always assigned **space first**, in order from **left to right**. These are followed by **percentage** values of the total space. Finally, **proportional** values are divided based upon what space is **left**.

What will be the Output?

```
<FRAMESET ROWS="*, 2*, *" COLS="2*, *">  
<FRAME SRC="">  
<FRAME SRC="">  
<FRAME SRC="">  
<FRAME SRC="">  
<FRAME SRC="">  
<FRAME SRC="">  
</FRAMESET>
```

FORMS

- Forms add the ability to web pages to not only provide the person viewing the document with dynamic information but also to obtain information from the person viewing it, and process the information.

Objectives:

Upon completing this section, you should be able to

1. Create a FORM.
2. Add elements to a FORM.
3. Define CGI (**Common Gateway Interface**).
4. Describe the purpose of a CGI Application.
5. Specify an action for the FORM.
 - Forms work in all browsers.
 - Forms are Platform Independent.

FORMS

- To insert a form we use the <FORM></FORM> tags. The rest of the form elements must be inserted in between the form tags.

```
<HTML> <HEAD>
```

```
<TITLE> Sample Form</TITLE>
```

```
</HEAD>
```

```
<BODY BGCOLOR="FFFFFF">
```

```
<FORM ACTION = http://www.xnu.com/formtest.asp>
```

```
<P> First Name: <INPUT TYPE="TEXT" NAME="fname"  
MAXLENGTH="50"> </P>
```

```
<P> <INPUT TYPE="SUBMIT" NAME="fsubmit1" VALUE="Send Info">  
</P>
```

```
</FORM>
```

```
</BODY> </HTML>
```

<FORM> element attributes

- **ACTION**: is the **URL** of the **CGI** (Common Gateway Interface) program that is going to accept the data from the form, process it, and send a response back to the browser.
- **METHOD**: **GET** (default) or **POST** specifies which **HTTP** method will be used to send the form's contents to the web server. The CGI application should be written to accept the data from either method.
- **NAME**: is a form name used by **VBScript** or **JavaScripts**.
- **TARGET**: is the target frame where the response page will show up.

Form Elements

- Form elements have properties: **Text** boxes, **Password** boxes, **Checkboxes**, **Option(Radio)** buttons, **Submit**, **Reset**, **File**, **Hidden** and **Image**.
- The properties are specified in the **TYPE** Attribute of the HTML element **<INPUT></INPUT>**.

Name:

Sami Ali

Student No.

123456789

Address:

Al al-Bayt University
CIS Department
Faculty of IT

City:

Amman

Amman

Irbed

Karak

is foreign?



Male:



Female:



Submit

Reset

Form Elements

<INPUT> Element's Properties

TYPE= Type of INPUT entry field.

NAME = Variable name passed to CGI application

VALUE= The data associated with the variable name to be passed to the CGI application

CHECKED= Button/box checked

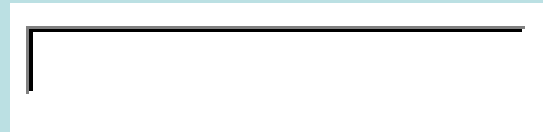
SIZE= Number of visible characters in text field

MAXLENGTH= Maximum number of characters accepted.

Text Box

- **Text boxes** : Used to provide input fields for text, phone numbers, dates, etc.

<INPUT TYPE= " TEXT " >



Browser will display

Textboxes use the following attributes:

- **TYPE**: text.
- **SIZE**: determines the size of the textbox in characters. **Default=20** characters.
- **MAXLENGTH** : determines the maximum number of characters that the field will accept.
- **NAME**: is the name of the variable to be sent to the CGI application.
- **VALUE**: will display its contents as the default value.

Example on Text Box

```
<TITLE>Form_Text_Type</TITLE>
</HEAD> <BODY>
<h1> <font color=blue>Please enter the following
  bioData</font></h1>
<FORM name="fome1" Method= " get " Action= " URL " >
First Name: <INPUT TYPE="TEXT" NAME="FName"
SIZE="15" MAXLENGTH="25"><BR>
Last Name: <INPUT TYPE="TEXT" NAME="LName"
SIZE="15" MAXLENGTH="25"><BR>
Nationality: <INPUT TYPE="TEXT" NAME="Country"
SIZE="25" MAXLENGTH="25"><BR>
The Phone Number: <INPUT TYPE="TEXT" NAME="Phone"
SIZE="15" MAXLENGTH="12"><BR>
</FORM> </BODY> </HTML>
```

Output

Form_Text_Type - Microsoft Internet Explorer

ملف تحرير عرض المفضلة أدوات تعليمات

« < > >> الخلف

« Links انتقال < >> عنوان C:\jdk\bin\tp01c7aa.html

Please enter the following bioData

First Name:

Last Name:

Nationality:

The Phone Number:

جهاز الكمبيوتر

Password

- **Password:** Used to allow entry of passwords.

<INPUT TYPE= " PASSWORD " >

Browser will display



Text typed in a password box is starred out in the browser display.

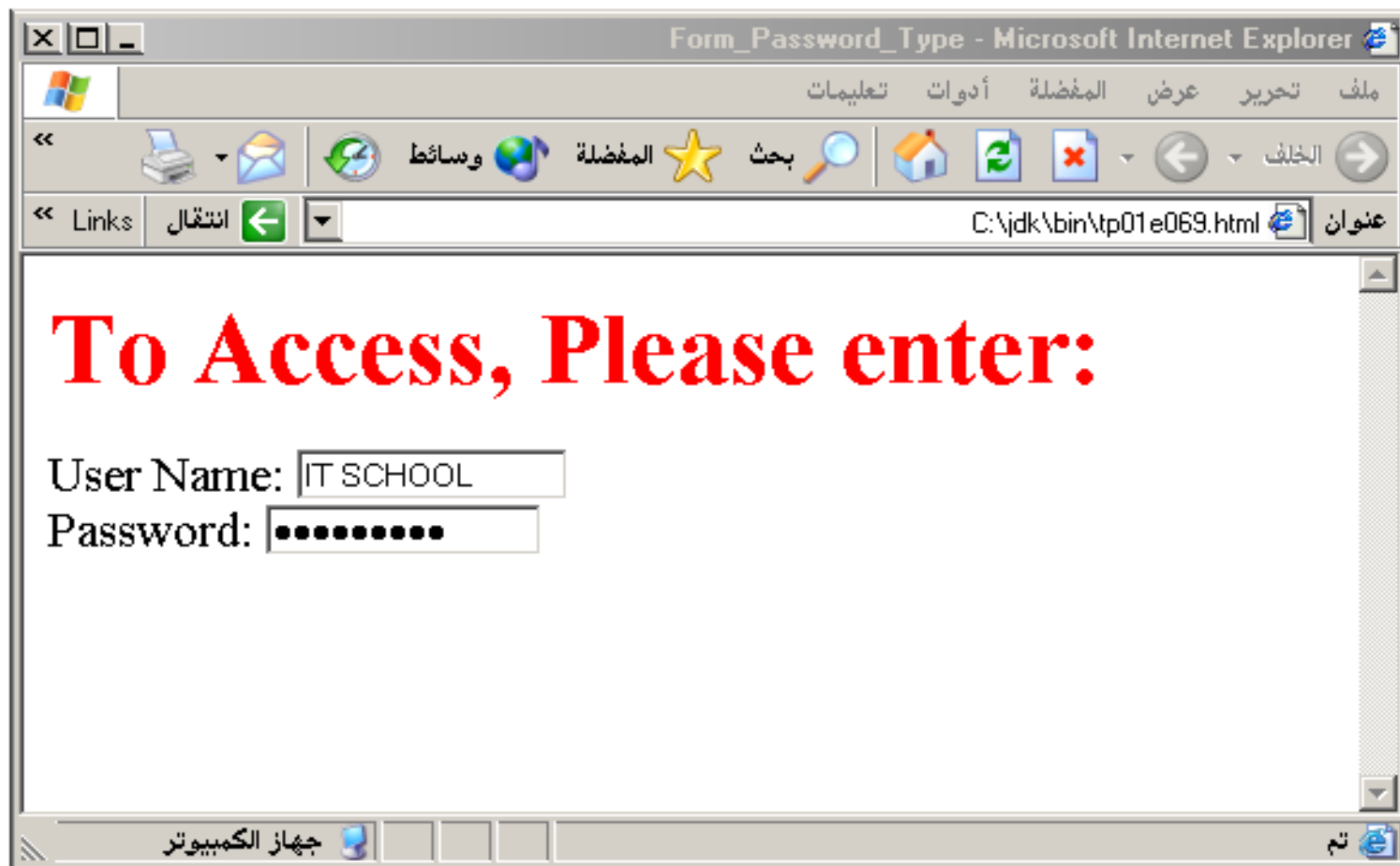
Password boxes use the following attributes:

- **TYPE:** password.
- **SIZE:** determines the size of the textbox in characters.
- **MAXLENGTH:** determines the maximum size of the password in characters.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually blank.

Example on Password Box

```
<HTML><HEAD>
<TITLE>Form_Password_Type</TITLE></HEAD>
<BODY>
<h1> <font color=red>To Access, Please
enter:</font></h1>
<FORM name="fome2" Action="url" method="get">
User Name: <INPUT TYPE="TEXT" Name="FName"
SIZE="15" MAXLENGTH="25"><BR>
Password: <INPUT TYPE="PASSWORD"
NAME="PWord" value="" SIZE="15"
MAXLENGTH="25"><BR>
</FORM></BODY> </HTML>
```

Output



Hidden

- **Hidden:** Used to send data to the CGI application that you don't want the web surfer to see, change or have to enter but is necessary for the application to process the form correctly.

<INPUT TYPE="HIDDEN">

Nothing is displayed in the browser.

Hidden inputs have the following attributes:

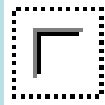
- **TYPE:** hidden.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually set a value expected by the CGI application.

Check Box

- **Check Box:** Check boxes allow the users to select more than one option.

<INPUT TYPE="CHECKBOX">

Browser will display

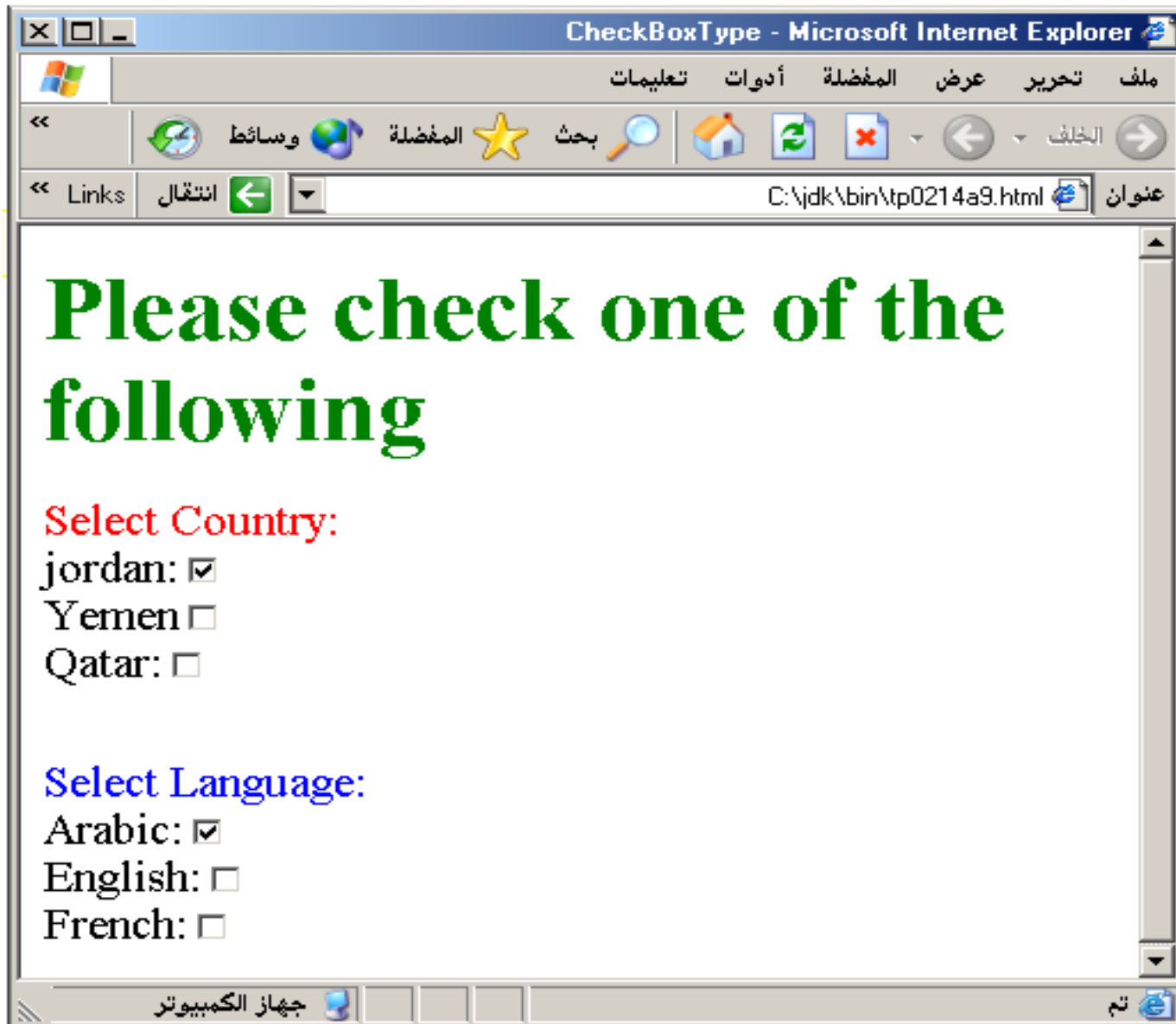


Checkboxes have the following attributes:

- **TYPE:** checkbox.
- **CHECKED:** is blank or CHECKED as the initial status.
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** is usually set to a value.


```
<HTML> <HEAD><TITLE>CheckBoxType</TITLE> </HEAD>
<BODY>
<h1> <font color=green>Please check one of the
following</font></h1>
<FORM name="fome3" Action="url" method="get">
<font color=red> Select Country: </font><BR>
jordan:<INPUT TYPE="CheckBox" Name="country"
CHECKED><BR>
Yemen<INPUT TYPE="CheckBox" Name="country"><BR>
Qatar:<INPUT TYPE="CheckBox" Name="country"><BR>
<BR>
<font color=blue>Select Language:</font><BR>
Arabic:<INPUT TYPE="CheckBox" Name="language"
CHECKED><BR> English:<INPUT TYPE="CheckBox"
Name="language"><BR>
French:<INPUT TYPE="CheckBox" Name="language">
<BR></FORM> </BODY></HTML>
```

Output



Radio Button

- **Radio Button:** Radio buttons allow the users to select only one option.

<INPUT TYPE="RADIO">

Browser will display



Radio buttons have the following attributes:

- **TYPE:** radio.
- **CHECKED:** is blank or CHECKED as the initial status. Only one radio button can be checked
- **NAME:** is the name of the variable to be sent to the CGI application.
- **VALUE:** usually has a set value.

```
<HTML> <HEAD><TITLE>CheckBoxType</TITLE> </HEAD>
<BODY>
<h1> <font color=green>Please check one of the
following</font></h1>
<FORM name="fome3" Action="url" method="get">
<font color=red> Select Country: </font><BR>
jordan:<INPUT TYPE= "RADIO" Name="country"
CHECKED><BR>
Yemen<INPUT TYPE="RADIO " Name="country"><BR>
Qatar:<INPUT TYPE="RADIO" Name="country"><BR>
<BR>
<font color=blue>Select Language:</font><BR>
Arabic:<INPUT TYPE="RADIO" Name="language"
CHECKED><BR> English:<INPUT TYPE=" RADIO "
Name="language"><BR>
French:<INPUT TYPE=" RADIO " Name="language">
<BR></FORM> </BODY></HTML>
```



```
<HTML><HEAD>
```

```
<TITLE>RADIOBox</TITLE> </HEAD>
```

```
<BODY>
```

Form #1:

```
<FORM>
```

```
<INPUT TYPE="radio" NAME="choice" VALUE="one"> Yes.
```

```
<INPUT TYPE="radio" NAME="choice" VALUE="two"> No.
```

```
</FORM>
```

```
<HR color=red size="10" >
```

Form #2:

```
<FORM>
```

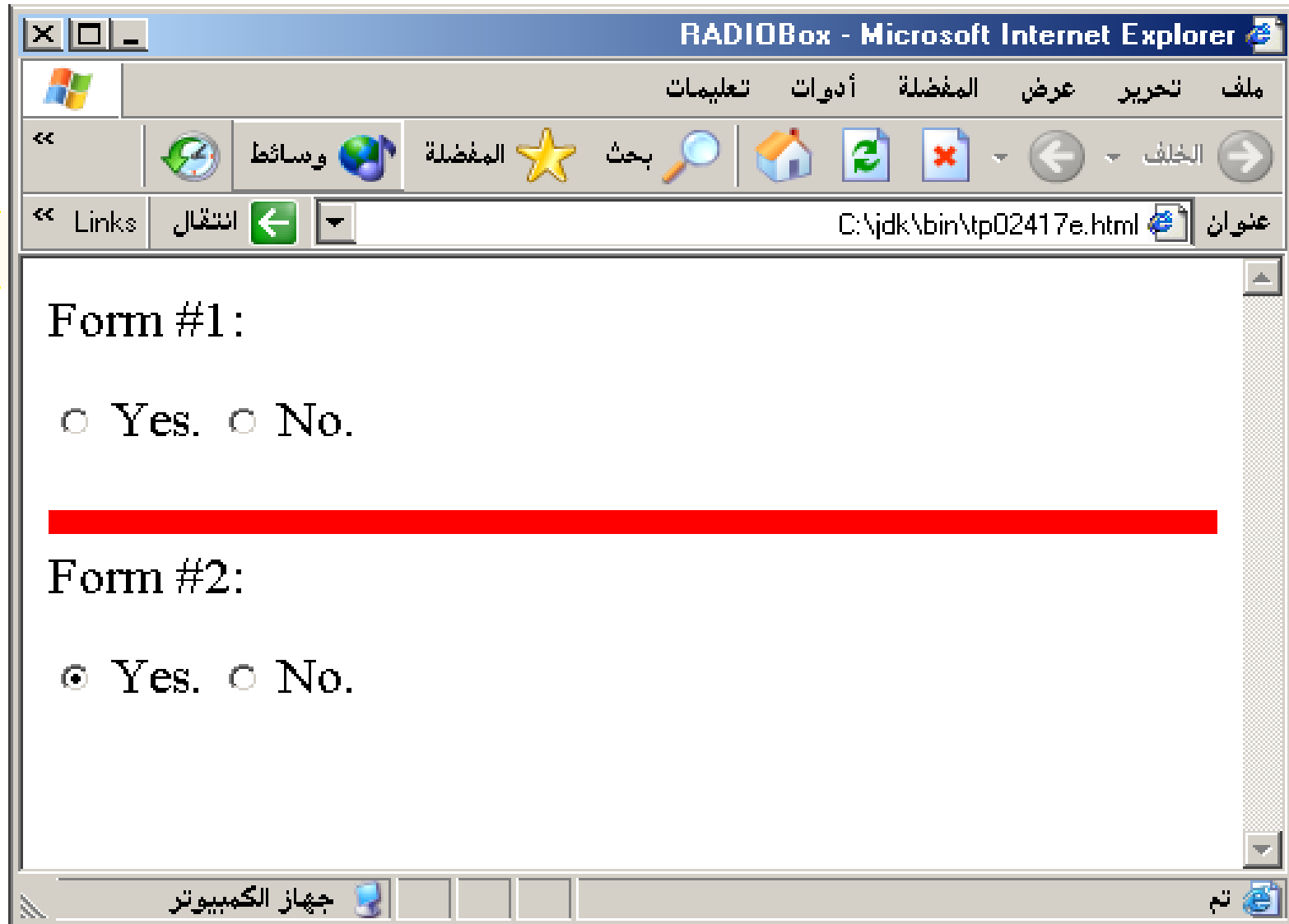
```
<INPUT TYPE="radio" NAME="choice" VALUE="three"  
CHECKED> Yes.
```

```
<INPUT TYPE="radio" NAME="choice" VALUE="four"> No.
```

```
</FORM>
```

```
</BODY></HTML>
```

Output



Push Button

- **Push Button:** This element would be used with JavaScript to cause an action to take place.

<INPUT TYPE="BUTTON">

Browser will display



Push Button has the following attributes:

- **TYPE:** button.
- **NAME:** is the name of the button to be used in scripting.
- **VALUE:** determines the text label on the button.

<DIV align=center>

<FORM>

<h1>Press Here to see a baby crying:

<INPUT TYPE="button"
VALUE="PressMe">

Click Here to see a baby shouting:

<INPUT TYPE="button" VALUE="ClickMe" >

Hit Here to see a baby eating:

<INPUT TYPE="button" VALUE="HitME" >

</FORM></DIV>



Submit Button

- **Submit:** Every set of Form tags requires a Submit button. This is the element causes the browser to send the names and values of the other elements to the CGI Application specified by the ACTION attribute of the FORM element.

<INPUT TYPE="SUBMIT">

The browser will display



Submit has the following attributes:

- **TYPE:** submit.
- **NAME:** value used by the CGI script for processing.
- **VALUE:** determines the text label on the button, usually Submit Query.

```
<FORM Action="URL" method="get">  
First Name: <INPUT TYPE="TEXT" Size=25  
name="firstName"><BR>  
Family Name: <INPUT TYPE="TEXT" Size=25  
name="LastName"><BR>  
<BR>  
<FONT Color=red>  
Press Here to submit the data:<BR>  
<INPUT TYPE="submit" VALUE="SubmitData " >  
</FORM>
```

C:\Documents and Settings\Khaled\My Documents\tp0118e7.html - Microsoft Internet Explorer

ملف تحرير عرض المفضلة أدوات تعليمات

انتقال < > الخلف < > عنوان C:\Documents and Settings\Khaled\My Documents\tp0118e7.html

First Name:

Family Name:

Press Here to submit the data:

جهاز الكمبيوتر

Reset Button

- **Reset:** It is a good idea to include one of these for each form where users are entering data. It allows the surfer to clear all the input in the form.

- **<INPUT TYPE="RESET">**

- Browser will display



-
- Reset buttons have the following attributes:
- **TYPE:** reset.
- **VALUE:** determines the text label on the button, usually Reset.

```
<FORM Action="URL" method="get">  
First Name: <INPUT TYPE="TEXT" Size=25  
name="firstName"> <BR>  
Family Name: <INPUT TYPE="TEXT" Size=25  
name="LastName"><BR>  
<BR>  
<FONT Color = red>  
<STRONG><font size=5>Press Here to submit  
the data:</font></STRONG><BR>  
<INPUT TYPE="submit" VALUE="SubmitData">  
<INPUT TYPE="RESET" VALUE="Reset">  
</FORM>
```

...C:\Documents and Settings\Khaled\My Documents\tp0125cb.html - Microsoft I

ملف تحرير عرض المفضلة أدوات تعليمات

« » الخلف

« Links انتقال عنوان C:\Documents and Settings\Khaled\My Documents\tp0125cb.html

First Name:

Family Name:

Press Here to submit the data:

SubmitData Reset

جهاز الكمبيوتر

٢٦

Image Submit Button

- **Image Submit Button:** Allows you to substitute an image for the standard submit button.

```
<INPUT TYPE="IMAGE" SRC="jordan.gif">
```

Image submit button has the following attributes:

- **TYPE:** Image.
- **NAME:** is the name of the button to be used in scripting.
- **SRC:** URL of the Image file.

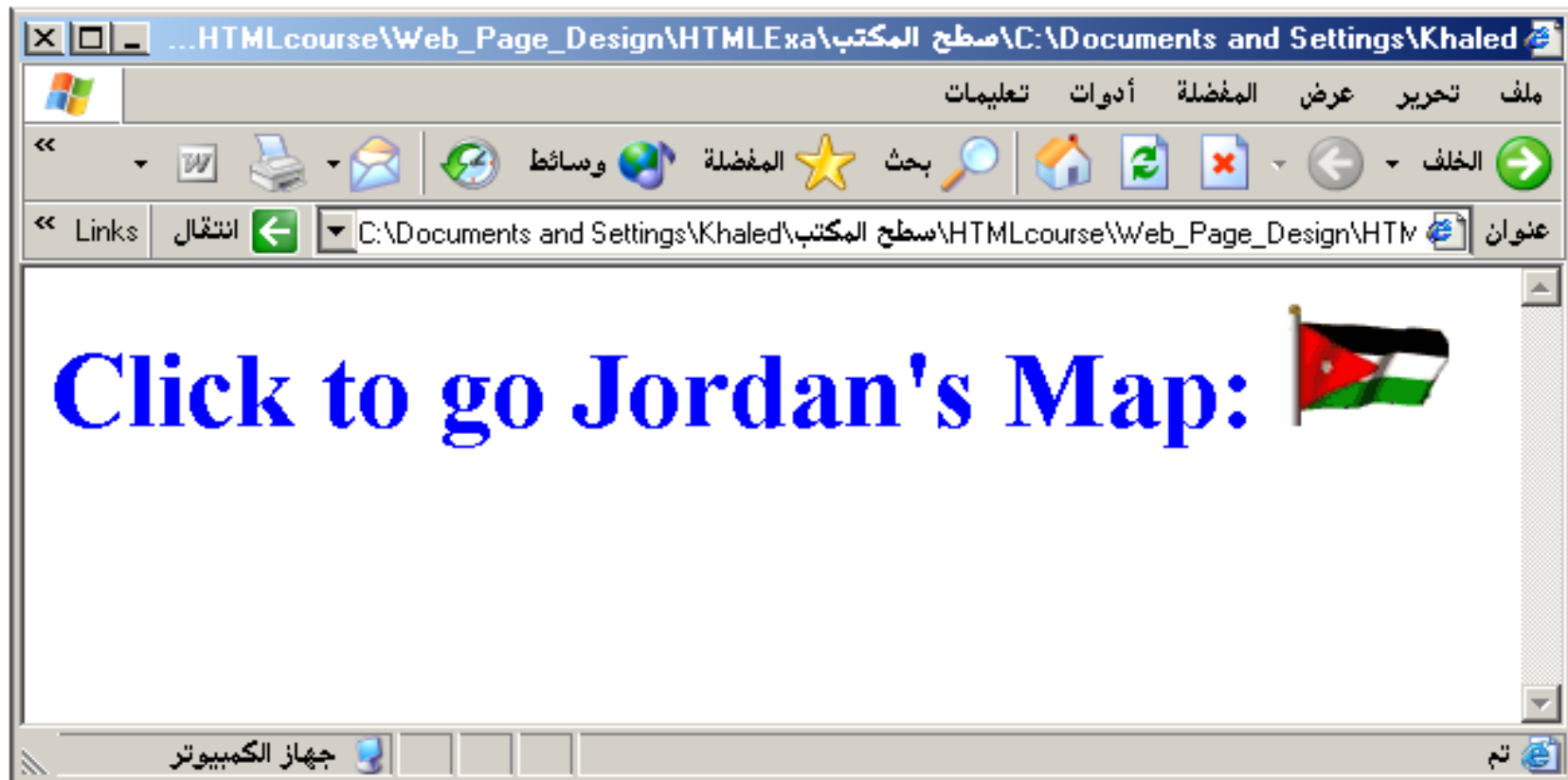
<form>

<H1>

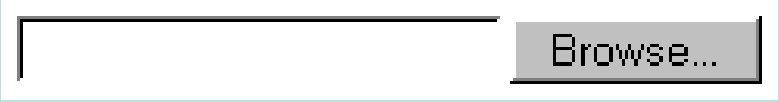
Click to go Jordan's Map:

<INPUT TYPE="IMAGE" SRC="jordan.gif">

</form>



File

- **File Upload:** You can use a file upload to allow surfers to upload files to your web server.
- **<INPUT TYPE="FILE">**
- Browser will display 
- File Upload has the following attributes:
 - **TYPE:** file.
 - **SIZE:** is the size of the text box in characters.
 - **NAME:** is the name of the variable to be sent to the CGI application.
 - **MAXLENGTH:** is the maximum size of the input in the textbox in characters.

```
<BODY bgcolor=lightblue>
```

```
<form>
```

```
<H3><font color=forestgreen>
```

```
Please attach your file here to for uploading to  
My <font color =red>SERVER...<BR>
```

```
<INPUT TYPE="File" name="myFile"  
size="30">
```

```
<INPUT TYPE="Submit" value="SubmitFile">
```

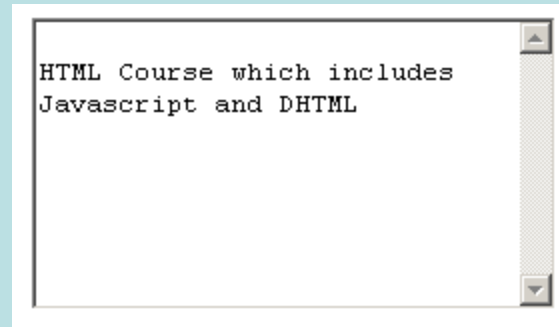
```
</form>
```

```
</BODY>
```

Other Elements used in Forms

- **<TEXTAREA></TEXTAREA>**: is an element that allows for free form text entry.

Browser will display



Textarea has the following attributes:

- **NAME**: is the name of the variable to be sent to the CGI application.
- **ROWS**: the number of rows to the textbox.
- **COLS**: the number of columns to the textbox.

```
<BODY bgcolor=lightblue>
```

```
<form>
```

```
<TEXTAREA COLS=40 ROWS=20
```

```
Name="comments" >
```

From observing the apathy of those about me during flag raising I concluded that patriotism if not actually on the decline is at least in a state of dormancy.

Written by Khaled Al-Fagih

```
</TEXTAREA>:
```

```
</form>
```

```
</BODY>
```

From observing the apathy of those about me during flag raising I concluded that patriotism if not actually on the decline is at least in a state of dormancy.
Written by Khaled Al-Fagih

10.6.1.2 The `wrap` attribute

Normally, text typed in the text area by the user is transmitted to the server exactly as typed, with lines broken only where the user pressed the Enter key. Since this is often not the desired action by the user, you can enable word wrapping within the text area. When the user types a line that is longer than the width of the text area, the browser automatically moves the extra text down to the next line, breaking the line at the nearest point between words in the line.

With the `wrap` attribute set to `virtual`, the text is wrapped within the text area for presentation to the user, but the text is transmitted to the server as if no wrapping had occurred, except where the user pressed the Enter key.

With the `wrap` attribute set to `physical`, the text is wrapped within the text area and is transmitted to the server as if the user had actually typed it that way. This is the most useful way to use word wrap, since the text is transmitted exactly as the user sees it in the text area.

To obtain the default action, set the `wrap` attribute to `off`.

As an example, consider the following 60 characters of text being typed into a 40-character-wide text area:

```
Word wrapping is a feature that makes life easier for users.
```

With `wrap=off`, the text area will contain one line and the user will have to scroll to the right to see all of the text. One line of text will be transmitted to the server.

With `wrap=virtual`, the text area will contain two lines of text, broken after the word "makes."
Only one line of text will be transmitted to the server: the entire line with no embedded newline characters.

With `wrap=physical`, the text area will contain two lines of text, broken after the word "makes."
Two lines of text will be sent to the server, separated by a newline character after the word "makes."

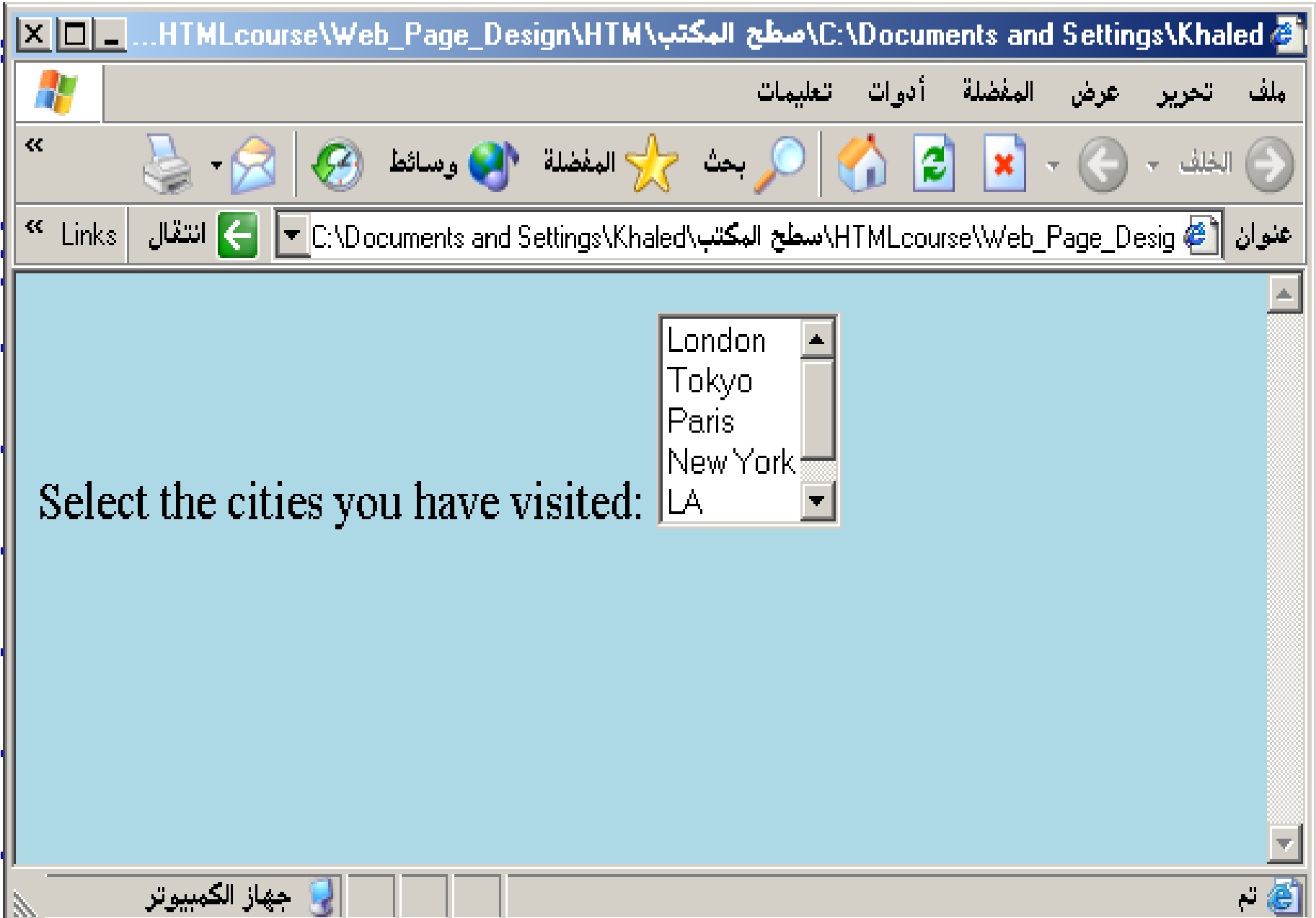
Other Elements used in Forms

- The two following examples are **<SELECT></SELECT>** elements, where the attributes are set differently.

The Select elements attributes are:

- **NAME**: is the name of the variable to be sent to the CGI application.
- **SIZE**: this sets the number of **visible** choices.
- **MULTIPLE**: the presence of this attribute signifies that the user can make multiple selections. By default only one selection is allowed.

```
<BODY bgcolor=lightblue>
<form>
Select the cities you have visited:
<SELECT name="list" size=5>
<option> London</option>
<option> Tokyo</option>
<option> Paris</option>
<option> New York</option>
<option> LA</option>
<option> KL</option>
</SELECT>
</form>
</BODY>
```



...HTMLcourse\Web_Page_Design\HTM\سطح المكتب\C:\Documents and Settings\Khaled

ملف تحرير عرض المفضلة أدوات تعليمات

« ووسائل المفضلة بحث الخلف

« Links انتقال عنوان C:\Documents and Settings\Khaled\سطح المكتب\HTMLcourse\Web_Page_Desig

Select the cities you have visited:

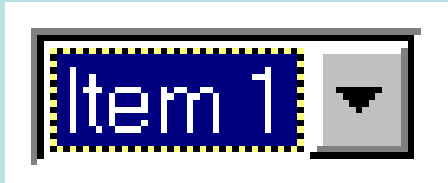
- London
- Tokyo
- Paris
- New York
- LA

جهاز الكمبيوتر

١٤:٠٠

Other Elements used in Forms

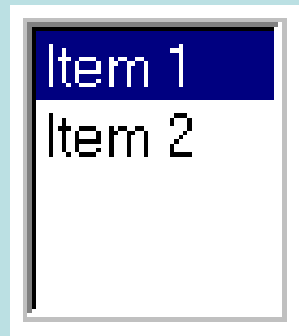
- **Drop Down List:**



- **Name:** is the name of the variable to be sent to the CGI application.
- **Size: 1.**

Other Elements used in Forms

- **List Box:**



- **Name:** is the name of the variable to be sent to the CGI application.
- **SIZE:** is greater than one.

Other Elements used in Forms

- **Option**

The list items are added to the **<SELECT>** element by inserting **<OPTION></OPTION>** elements.

The Option Element's attributes are:

- **SELECTED**: When this attribute is present, the option is selected when the document is initially loaded. **It is an error for more than one option to be selected.**
- **VALUE**: Specifies the value the variable named in the select element.

</HEAD>

<BODY>

<h2>What type of Computer do you have?<h2>

<FORM>

<SELECT NAME="ComputerType" size=4>

<OPTION value="IBM" SELECTED> IBM</OPTION>

<OPTION value="INTEL"> INTEL</OPTION>

<OPTION value=" Apple"> Apple</OPTION>

<OPTION value="Compaq"> Compaq</OPTION>

</SELECT>

</FORM></BODY></HTML>



What type of Computer do you have?

IBM
INTEL
Apple
Compaq

```
<HEAD> <TITLE>SELECT with Mutiple </TITLE>
</HEAD>
```

```
<BODY>
```

```
<h2><font color=blue>What type of Computer do you
have?</font><h2>
```

```
<FORM>
```

```
<SELECT NAME="ComputerType" size=5 multiple>
```

```
<OPTION value="IBM" > IBM</OPTION>
```

```
<OPTION value="INTEL"> INTEL</OPTION>
```

```
<OPTION value=" Apple"> Apple</OPTION>
```

```
<OPTION value="Compaq" SELECTED>
Compaq</OPTION>
```

```
<OPTION value=" other"> Other</OPTION>
```

```
</SELECT>
```

```
</FORM></BODY></HTML>
```

What type of Computer do you have?

- IBM
- INTEL
- Apple
- Compaq
- Other

There are eleven different types of form elements:

Button

Checkbox

FileUpload

Hidden

Password

Radio

Reset object

Select object

Submit object

Text

Textarea

Cascading Style Sheets

.

CSS

- ▶ CSS stands for **Cascading Style Sheets**
- ▶ Styles define **how to display** HTML elements
- ▶ Styles were added to HTML 4.0 **to solve a problem**
- ▶ **External Style Sheets** can save a lot of work

Selector

Declaration

Declaration

h1

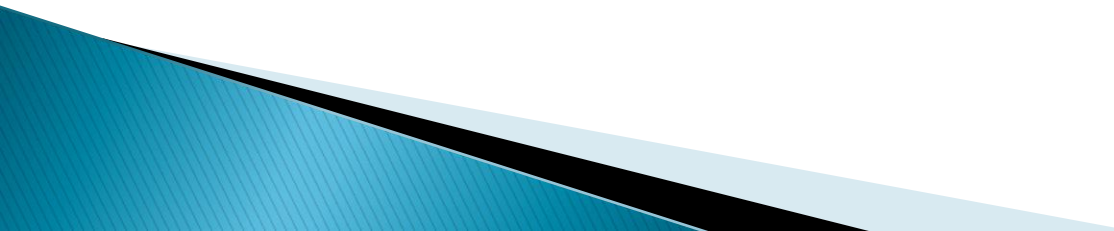
```
{ color:blue; font-size:12px; }
```

Property

Value

Property

Value

- ▶ A CSS rule has two main parts: a selector, and one or more declarations:
 - ▶ The selector is normally the HTML element you want to style.
 - ▶ Each declaration consists of a property and a value.
 - ▶ The property is the style attribute you want to change. Each property has a value.
- 

Internal Stylesheet

First we will explore the internal method.

This way you are simply placing the CSS code within the `<head></head>` tags of each HTML file you want to style with the CSS. The format for this is shown in the example below

- ▶ `<head>`
- ▶ `<title><title>`
- ▶ `<style type="text/css">`
CSS Content Goes Here
- ▶ `</style>`
- ▶ `</head>`
- ▶ `<body>`

With this method each HTML file contains the CSS code needed to style the page. Meaning that any changes you want to make to one page, will have to be made to all. This method can be good if you need to style only one page, or if you want different pages to have varying styles.

```
p {color:red;text-align:center;}
```

- ▶ `<html>`
- ▶ `<head>`
- ▶ `<style type="text/css">`
- ▶ `h1 {color:red;}`
- ▶ `h2 {color:blue;}`
- ▶ `p {color:green;}`
- ▶ `</style>`
- ▶ `</head>`
- ▶ `<body>`
- ▶ `<h1>All header 1 elements will be red</h1>`
- ▶ `<h2>All header 2 elements will be blue</h2>`
- ▶ `<p>All text in paragraphs will be green.</p>`
- ▶ `</body>`
- ▶ `</html>`

JavaScript Introduction

JavaScript was released by Netscape and Sun Microsystems in 1995. However, JavaScript is not the same thing as Java.

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Put a JavaScript into an HTML page

To insert a JavaScript into an HTML page, we use the `<script>` tag. Inside the `<script>` tag we use the `type` attribute to define the scripting language.

```
<script type="text/JavaScript">  
...some JavaScript  
</script>
```

Installation is not required, nor do you have to torturously work through any odd library path configurations. JavaScript works, straight out of the box and in most web browsers, including the big four: Firefox, Internet Explorer, Opera, and Safari. All you need to do is add a scripting block, and you're in business.

The **document.write** command is a standard JavaScript command for writing output to a page.

By entering the `document.write` command between the `<script>` and `</script>` tags, the browser will recognize it as a JavaScript command and execute the code line. In this case the browser will write Hello World! to the page

```
<script type="text/javascript">
document.write("Hello World!");
</script>
```

The example below shows how to add HTML tags to the JavaScript:

```
document.write("<h1>Hello World!</h1>");
```

JavaScript is a sequence of statements to be executed by the browser.

JavaScript is Case Sensitive

Unlike HTML, JavaScript is case sensitive - therefore watch your capitalization closely when you write JavaScript statements, create or call variables, objects and functions.

JavaScript Statements

A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do. This JavaScript statement tells the browser to write "Hello Dolly" to the web page.

It is normal to add a semicolon at the end of each executable statement. Most people think this is a good programming practice, and most often you will see this in JavaScript examples on the web.

The semicolon is optional (according to the JavaScript standard), and the browser is supposed to interpret the end of the line as the end of the statement. Because of this you will often see examples without the semicolon at the end.

JavaScript code (or just JavaScript) is a sequence of JavaScript statements. Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks. Blocks start with a left curly bracket {, and ends with a right curly bracket }. The purpose of a block is to make the sequence of statements execute together.

This example will write a heading and two paragraphs to a web page

```
<script type="text/javascript">
{
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
}
</script>
```

The example above is not very useful. It just demonstrates the use of a block. Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

JavaScript Variables

Variables in JavaScript are much like those in any other language; you use them to hold values in such a way that the values can be explicitly accessed in different places in the code. Each has an identifier that is unique to the scope of use, consisting of any combination of letters, digits, underscores, and dollar signs. An identifier

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Table JavaScript keyword

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	public

Creating JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var** keyword:

```
var x;  
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

However, you can also assign values to the variables when you declare them:

```
var x=5;  
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

The following snippet of code assigns a string literal containing a line-terminator escape sequence to a variable. When the string is used in a dialog window, the escape sequence, \n, is interpreted literally, and a newline is published:

```
var string_value = "This is the first line\nThis is the second line";
```

This results in:

This is the first line

This is the second line

Redeclaring JavaScript Variables

If you re declare a JavaScript variable, it will not lose its original value.

```
Var x=5;  
var x;
```

After the execution of the statements above, the variable x will still have the value of 5. The value of x is not reset (or cleared) when you redeclare it.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;  
z=y+5;
```

You will learn more about the operators that can be used in the next chapter of this tutorial.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3

*	Multiplication	$x=y*2$	$x=10$
/	Division	$x=y/2$	$x=2.5$
%	Modulus (division remainder)	$x=y\%2$	$x=1$
++	Increment	$x=++y$	$x=6$
--	Decrement	$x=--y$	$x=4$

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that $x=10$ and $y=5$, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	$x=y$		$x=5$
+=	$x+=y$	$x=x+y$	$x=15$
-=	$x-=y$	$x=x-y$	$x=5$
=	$x=y$	$x=x*y$	$x=50$
/=	$x/=y$	$x=x/y$	$x=2$
%=	$x\%=y$	$x=x\%y$	$x=0$

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";  
txt2="nice day";  
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";  
txt2="nice day";  
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"

Adding Strings and Numbers

The rule is: If you add a number and a string, the result will be a string!

Example

```
x=5+5;  
document.write(x);
```

```
x="5"+"5";  
document.write(x);
```

```
x=5+"5";  
document.write(x);
```

```
x="5"+5;  
document.write(x);
```

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Given that **x=5**, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x=== "5" is false

!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	And	(x < 10 && y > 1) is true
	Or	(x==5 y==5) is false
!	Not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

variablename=(condition)?value1:value2

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if the condition is true and another code if the condition is false

- **if...else if...else statement** - use this statement to select one of many blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
If (condition)
{
code to be executed if condition is true }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10
var d=new Date();
var time=d.getHours();
if (time<10)
{
document.write("<b>Good morning</b>"); }
</script>
```

Notice that there is no `..else..` in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

If...else Statement

Use the if...else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
{
code to be executed if condition is true
}
else
{
```

```
code to be executed if condition is not true
}
```

If...else if...else Statement

Use the if...else if...else statement to select one of several blocks of code to be executed.

Syntax

```
if (condition1)
{
  code to be executed if condition1 is true
}
else if (condition2)
{
  code to be executed if condition2 is true
}
else
{
  code to be executed if condition1 and condition2 are not true
}
```

The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

JavaScript Popup Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
Alert('sometext');
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

JavaScript Functions

A function will be executed by an event or by a call to the function. To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!"); }
</script>
</head>
<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```

If the line: alert("Hello world!!") in the example above had not been put within a function, it would have been executed as soon as the page was loaded. Now, the script is not executed before a user hits the input button. The function displaymessage() will be executed if the input button is clicked.

You will learn more about JavaScript events in the JS Events chapter.

The return Statement

The return statement is used to specify the value that is returned from the function. So, functions that are going to return a value must use the return statement. The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script></body> </html>
```

The Lifetime of JavaScript Variables

If you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.

If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are <html>

Example

```
<head>

<script type="text/javascript">

function myfunction(txt)

{ alert(txt);}
```



```
</script> </head> <body>

<form>

<input type="button" onclick="myfunction('Hello')" value="Call
function">

</form>

<p>By pressing the button above, a function will be called with "Hello"
as a parameter. The function will alert the parameter.</p>

</body> </html>
```

JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
{
  code to be executed
}
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
{
  code to be executed
}
while (var<=endvalue);
```

The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

The continue Statement

The continue statement will break the current loop and continue with the next value

What is an Array?

An array is a special variable, which can hold more than one value, at a time. If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array! An array can hold all your variable values under a single name. And you can access the values by referring to the array name. Each element in the array has its own ID so that it can be easily accessed.

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
{
  code to be executed
}
```

Create an Array

An array can be defined in three ways. The following code creates an Array object called myCars

1:

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";      // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

2:

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed
array
```

3:

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.

Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0. The following code line:

```
document.write(myCars[0]);  
will result in the following output: Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);  
will result in the following output: Opel
```

Arrays - concat()

```
var parents = ["Jani", "Tove"];  
var children = ["Cecilie", "Lone"];  
var family = parents.concat(children);  
document.write(family);
```

Insert Special Characters

The backslash (\) is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Look at the following JavaScript code:

```
var txt="We are the so-called "Vikings" from the north.";
```

document.write(txt);

In JavaScript, a string is started and stopped with either single or double quotes. This means that the string above will be chopped to: We are the so-called

To solve this problem, you must place a backslash (\) before each double quote in "Viking". This turns each double quote into a string literal:

var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);

JavaScript will now output the proper text string: We are the so-called "Vikings" from the north.

Here is another example:

document.write ("You \& I are singing!");

The example above will produce the following output:

You & I are singing!

The table below lists other special characters that can be added to a text string with the backslash sign:

Code	Outputs
\'	single quote
\"	Double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	Tab
\b	backspace
\f	form feed

String object

1- Return the length of a string

```
var txt = "Hello World!"; document.write(txt.length);
```

2- Style strings

```
var txt = "Hello World!";
```

```
document.write("<p>Big: " + txt.big() + "</p>");
```

```
document.write("<p>Bold: " + txt.bold() + "</p>");
```

```
document.write("<p>Italic: " + txt.italics() + "</p>");
```

3- The toLowerCase() and toUpperCase() methods

```
var txt="Hello World!";
```

```
document.write(txt.toLowerCase() + "<br />");
```

```
document.write(txt.toUpperCase());
```

4- The match() method

```
var str="Hello world!";
```

```
document.write(str.match("world") + "<br />");
```

```
document.write(str.match("World") + "<br />");
```

5- Replace characters in a string - replace()

```
var str="Visit Microsoft!";
```

```
document.write(str.replace("Microsoft"," Schools"));
```

6- The indexOf() method: How to return the position of the first found occurrence of a specified value in a string.

```
var str="Hello world!";
```

```
document.write(str.indexOf("d") + "<br />");
```

```
document.write(str.indexOf("world"));
```

The try...catch Statement

The try...catch statement allows you to test a block of code for errors. The try block contains the code to be run, and the catch block contains the code to be executed if an error occurs.

Syntax

```
Try  
{  
  //Run some code here  
}  
catch(err)  
{  
  //Handle errors here  
}
```

Note that try...catch is written in lowercase letters. Using uppercase letters will generate a JavaScript error!

Create a Date Object

The Date object is used to work with dates and times. Date objects are created with the Date() constructor. There are four ways of instantiating a date:

```
new Date() // current date and time
```

Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time.

All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds.

Some examples of instantiating a date:

```
today = new Date()  
d1 = new Date("October 13, 1975 11:13:00")  
d2 = new Date(79,5,24)  
d3 = new Date(79,5,24,11,33,0)
```

Set Dates

We can easily manipulate the date by using the methods available for the Date object. In the example below we set a Date object to a specific date (14th January 2010):

```
var myDate=new Date();  
myDate.setFullYear(2010,0,14);
```

And in the following example we set a Date object to be 5 days into the future:

```
var myDate=new Date();  
myDate.setDate(myDate.getDate()+5);
```

Note: If adding five days to a date shifts the month or year, the changes are handled automatically by the Date object itself!

Compare Two Dates

The Date object is also used to compare two dates. The following example compares today's date with the 14th January 2010:

```
var myDate=new Date();  
myDate.setFullYear(2010,0,14);  
var today = new Date();  
if (myDate>today)  
{  
  alert("Today is before 14th January 2010");  
}  
else  
{  
  alert("Today is after 14th January 2010");  
}
```

JavaScript Math Object

round()

How to use round().


```
document.write(Math.round(0.60) + "<br />");
```

random()

How to use random() to return a random number between 0 and 1.

```
//return a random number between 0 and 1
```

```
document.write(Math.random() + "<br />");
```

```
//return a random integer between 0 and 10
```

```
document.write(Math.floor(Math.random()*11));
```

max()

How to use max() to return the number with the highest value of two specified numbers.

```
document.write(Math.max(0,150,30,20,38) + "<br />");
```

```
document.write(Math.max(-5,10) + "<br />");
```

min()

How to use min() to return the number with the lowest value of two specified numbers.

```
document.write(Math.min(-5,-10) + "<br />");
```

```
document.write(Math.min(1.5,2.5));
```

Method of object

Function	Description	Returned Values
getDate()	Day of the month	1-31
getDay()	Day of the week (integer)	0-6
getFullYear()	Year (full four digit)	1900+
getHours()	Hour of the day (integer)	0-23
getMilliseconds()	Milliseconds (since last second)	0-999
getMinutes()	Minutes (since last hour)	0-59
getMonth()	Month	0-11
getSeconds()	Seconds (since last minute)	0-59
getTime()	Number of milliseconds since 1	

	January 1970	
getTimezoneOffset()	Difference between local time and GMT in minutes	0-1439
getYear()	Year	0-99 for years between 1900-1999 Four digit for 2000+
parse()	Returns the number of milliseconds since midnight 1 January 1970 for a given date and time string passed to it.	
setDate()	Sets the day, given a number between 1-31	Date in milliseconds
setFullYear()	Sets the year, given a four digit number	Date in milliseconds
setHours()	Sets the hour, given a number between 0-23	Date in milliseconds
setMilliseconds()	Sets the milliseconds, given a number	Date in milliseconds
setMinutes()	Sets the minutes, given a number between 0-59	Date in milliseconds
setMonth()	Sets the month, given a number between 0-11	Date in milliseconds
setSeconds()	Sets the seconds, given a number between 0-59	Date in milliseconds
setTime()	Sets the date, given the number of milliseconds since 1 January 1970	Date in milliseconds
setYear()	Sets the year, given either a two digit or four digit number	Date in milliseconds
toGMTString() toUTCString()	GMT date and time as a string	day dd mmm yyyy hh:mm:ss GMT
toLocaleString()	Local date and time as a string	Depends on operating system, locale, and browser
toString()	Local date and time as a string	Depends on operating system, locale, and browser
UTC()	Returns the number of milliseconds since 1 January 1970 for a given date in year, month, day (and optionally, hours, minutes, seconds, and milliseconds)	Date in milliseconds
valueOf()	Number of milliseconds since 1 January 1970	Date in milliseconds

JavaScript Form Validation

There's nothing more troublesome than receiving orders, guestbook entries, or other form submitted data that are incomplete in some way. You can avoid these headaches once and for all with JavaScript's amazing way to combat bad form data with a technique called "form validation".

JavaScript `document.getElementById`

JavaScript `getElementById`

Have you ever tried to use JavaScript to do some form validation? Did you have any trouble using JavaScript to grab the value of your text field? There's an easy way to access any HTML element, and it's through the use of *id* attributes and the *getElementById* function.

If you want to quickly access the value of an HTML input give it an *id* to make your life a lot easier. This small script below will check to see if there is any text in the text field "myText". The argument that *getElementById* requires is the *id* of the HTML element you wish to utilize.

```
<script type="text/javascript">
function notEmpty(){
    var myTextField = document.getElementById('myText');
    if(myTextField.value != "")
        alert("You entered: " + myTextField.value)
    else
        alert("Would you please enter some text?")
}
</script>
<input type='text' id='myText' />
<input type='button' onclick='notEmpty()' value='Form Checker' />
```

The `innerHTML` property can be used to modify your document's HTML on the fly.

When you use `innerHTML`, you can change the page's content without refreshing the page. This can make your website feel quicker and more responsive to user input.

The `innerHTML` property is used along with `getElementById` within your JavaScript code to refer to an HTML element and change its contents.

The `innerHTML` property is not actually part of the official DOM specification. Despite this, it is supported in all major browsers, and has had widespread use across the web for many years. DOM, which stands for Document Object Model, is the hierarchy that you can use to access and manipulate HTML objects from within your JavaScript.

The innerHTML Syntax

The syntax for using `innerHTML` goes like this:

```
document.getElementById('{ID of  
element}').innerHTML = '{content}';
```

In this syntax example, `{ID of element}` is the ID of an HTML element and `{content}` is the new content to go into the element.

Basic innerHTML Example

Here's a basic example to demonstrate how `innerHTML` works.

This code includes two functions and two buttons. Each function displays a different message and each button triggers a different function.

In the functions, the `getElementById` refers to the HTML element by using its ID. We give the HTML element an ID of "myText" using `id="myText"`.

So in the first function for example, you can see that

```
document.getElementById('myText').innerHTML =  
'Thanks!'; is setting the innerHTML of the "myText" element to  
"Thanks!".
```

Code:

```
<script type="text/javascript">  
function Msg1(){  
    document.getElementById('myText').innerHTML = 'Thanks!';  
}  
function Msg2(){  
    document.getElementById('myText').innerHTML = 'Try message 1  
again...';  
}  
</script>  
<input type="button" onclick="Msg1()" value="Show Message 1" />  
<input type="button" onclick="Msg2()" value="Show Message 2" />  
<p id="myText"></p>
```

Result:



Example 2: `innerHTML` With User Input

Here's an example of using `innerHTML` with a text field. Here, we display whatever the user has entered into the input field.

Code:

```
<script type="text/javascript">  
function showMsg(){  
    var userInput = document.getElementById('userInput').value;  
    document.getElementById('userMsg').innerHTML = userInput;  
}  
</script>
```

```
<input type="input" maxlength="40" id="userInput"
  onkeyup="showMsg()" value="Enter text here..." />
<p id="userMsg"></p>
```

Result:



Example 3: Formatting with getElementById

In this example, we use the getElementById property to detect the color that the user has selected. We can then use style.color to apply the new color. In both cases, we refer to the HTML elements by their ID (using getElementById.)

Code:

```
<script type="text/javascript">
function changeColor(){
  var newColor = document.getElementById('colorPicker').value;
  document.getElementById('colorMsg').style.color = newColor;
}
</script>
<p id="colorMsg">Choose a color...</p>
<select id="colorPicker" onchange="JavaScript:changeColor()">
<option value="#000000">Black</option>
<option value="#000099">Blue</option>
<option value="#990000">Red</option>
<option value="#009900">Green</option>
</select>
```

Result:



Choose a color...

String Search Function

This string function takes a regular expression and then examines that string to see if there are any matches for that expression. If there is a match, it will return the position in the string where the match was found. If there isn't a match, it will return -1. We won't be going into great depth about regular expressions, but we will show you how to search for words in a string.

Search Function Regular Expression

The most important thing to remember when creating a regular expression is that it must be surrounded with slashes */regular expression/*. With that knowledge let's search a string to see if a common name "Alex" is inside it.

```
<script type="text/javascript">
var myRegExp = /Alex/;
var string1 = "Today John went to the store and talked with Alex.";
var matchPos1 = string1.search(myRegExp);
```

```

if(matchPos1 != -1)
    document.write("There was a match at position " +
matchPos1);
else
    document.write("There was no match in the first string");
</script>

```

The quickest way to check if an input's string value is all numbers is to use a regular expression `/^[0-9]+$/` that will only match if the string is all numbers and is at least one character long.

JavaScript Code:

// If the element's string matches the regular expression it is all numbers.

```

var numericExpression = /^[0-9]+$/;

```

What we're doing here is using JavaScript existing framework to have it do all the hard work for us. Inside each string is a function called `match` that you can use to see if the string matches a certain regular expression. We accessed this function like so: `elem.value.match(expressionhere)`. We wanted to see if the input's string was all numbers so we made a regular expression to check for numbers `[0-9]` and stored it as `numericExpression`.

We then used the `match` function with our regular expression. If it is numeric then `match` will return `true`, making our `if` statement pass the test and our function `isNumeric` will also return `true`. However, if the expression fails because there is a letter or other character in our input's string then we'll display our `helperMsg` and return `false`.

```

if(elem.value.match(numericExpression)){
    return true;
else return false;

```

charAt

`charAt()` gives you the character at a certain position. For instance, when you do

```

var b = 'I am a JavaScript hacker.'
document.write(b.charAt(5))

```

split

`split()` does not work in Netscape 2 and Explorer 3.

`split()` is a specialized method that you need sometimes. It allows you to split a string at the places of a certain character. You must put the result in an array, not in a simple variable. Let's split `b` on the spaces.

```
var b = 'I am a JavaScript hacker.'  
var temp = new Array();  
temp = b.split(' ');
```

Now the string has been split into 5 strings that are placed in the array temp. The **spaces themselves are gone.**

```
temp[0] = 'I';  
temp[1] = 'am';  
temp[2] = 'a';  
temp[3] = 'JavaScript';  
temp[4] = 'hacker.';
```

Changing HTML with innerHTML

You can also insert HTML into your elements in the exact same way. Let's say we didn't like the text that was displayed in our paragraph and wanted to update it with some color. The following code will take the old black text and make it bright white. The only thing we're doing different here is inserting the html element span to change the color.

JavaScript Code:

```
var oldHTML =  
document.getElementById('para').innerHTML;
```

Form Validation - Checking for All Letters

This function will be identical to isNumeric except for the change to the regular expression we use inside the match function. Instead of checking for numbers we will want to check for all letters.

If we wanted to see if a string contained only letters we need to specify an expression that allows for both lowercase and uppercase letters: `/^[a-zA-Z]+$/`.

JavaScript Code:

```
// If the element's string matches the regular expression it is all letters  
function isAlphabet(elem, helperMsg){  
    var alphaExp = /^[a-zA-Z]+$/;  
    if(elem.value.match(alphaExp)){  
        return true;  
    }else{  
        alert(helperMsg);  
        elem.focus();  
        return false;  
    }  
}
```

Form Validation - Restricting the Length

Being able to restrict the number of characters a user can enter into a field is one of the best ways to prevent bad data. For example, if you know that the zip code field should only be 5 numbers you know that 2 numbers is not sufficient.

Below we have created a lengthRestriction function that takes a text field and two numbers. The first number is the minimum number of characters and the second is the maximum number of a characters the input can be. If you just want to specify an exact number then send the same number for both minimum and maximum.

JavaScript Code:

```
function lengthRestriction(elem, min, max){
    var uInput = elem.value;
    if(uInput.length >= min && uInput.length <= max){
        return true;
    }else{
        alert("Please enter between " +min+ " and " +max+ "
characters");
        elem.focus();
        return false;
    }
}
```

Form Validation - Email Validation

And for our grand finale we will be showing you how to check to see if a user's email address is valid. Every email is made up for 5 parts:

A combination of letters, numbers, periods, hyphens, plus signs, and/or underscores

The at symbol @

A combination of letters, numbers, hyphens, and/or periods

A period

The top level domain (com, net, org, us, gov, ...)

Valid Examples:

bobby.jo@filltank.net

jack+jill@hill.com

the-stand@steven.king.com

Invalid Examples:

@deleted.net - no characters before the @

free!dom@bravehe.art - invalid character !

shoes@need_shining.com - underscores are not allowed in the domain name

The regular expression to check for all of this is a little overkill and beyond the scope of this tutorial to explain thoroughly. However, test it out and you'll see that it gets the job done

External JavaScript File

You can place all your scripts into an external file (with a .js extension), then link to that file from within your HTML document. This is handy if you need to use the same scripts across multiple HTML pages, or a whole website.

To link to an external JavaScript file, you add a `src` attribute to your HTML `script` tag and specify the location of the external JavaScript file.

Linking to an external JavaScript file

```
<script type="text/javascript"
src="external_javascript.js"></script>
```

Contents of your external JavaScript file

The code in your **.js file** should be no different to the code you would normally have placed in between the script tags. But remember, you don't need to create script tag again - it is already on the HTML page calling the external file!

In the previous lesson, we used an *event handler* to trigger off a call to our function. There are 18 event handlers that you can use to link your HTML elements to a piece of JavaScript.


When you write a JavaScript function, you will need to determine when it will run. Often, this will be when a user does something like click or hover over something, submit a form, double clicks on something etc.

These are examples of *events*.

Using JavaScript, you can respond to an event using event handlers. You can attach an event handler to the HTML element for which you want to respond to when a specific event occurs.

For example, you could attach JavaScript's `onmouseover` event handler to a button and specify some JavaScript to run whenever this event occurs against that button.

The HTML 4 specification refers to these as intrinsic events and defines 18 as listed below:

 Event Handler	Event that it handles
<code>onBlur</code>	User has left the focus of the object. For example, they clicked away from a text field that was previously selected.
<code>onChange</code>	User has changed the object, then attempts to leave that field (i.e. clicks elsewhere).
<code>onClick</code>	User clicked on the object.

Event Handler	Event that it handles
onDblClick	User clicked twice on the object.
onFocus	User brought the focus to the object (i.e. clicked on it/tabbed to it)
onKeyDown	A key was pressed over an element.
onKeyUp	A key was released over an element.
onKeyPress	A key was pressed over an element then released.
onLoad	The object has loaded.
onMouseDown	The cursor moved over the object and mouse/pointing device was pressed down.
onMouseup	The mouse/pointing device was released after being pressed down.
onMouseover	The cursor moved over the object (i.e. user hovers the mouse over the object).
onMouseMove	The cursor moved while hovering over an object.
onMouseout	The cursor moved off the object
onReset	User has reset a form.
onSelect	User selected some or all of the contents of the object. For example, the user selected some text within a text field.
onSubmit	User submitted a form.
onUnload	User left the window (i.e. user closes the browser window).

The events in the above table provide you with many opportunities to trigger some JavaScript from within your HTML code.

I encourage you to bookmark this page as a reference - later on you may need a reminder of which events you can use when solving a particular coding issue.

Sometimes, you may need to call some JavaScript from within a link. Normally, when you click a link, the browser loads a new page (or refreshes the same page).

This might not always be desirable. For example, you might only want to dynamically update a form field when the user clicks a link.

JavaScript "On Double Click"

You could just have easily used another event to trigger the same JavaScript. For example, you could run JavaScript only when the double

clicks the HTML element. We can do this using the `onDbLcLick` event handler.

Code:

```
<input type="button" onDbLcLick="alert('Hey,
remember to tell your friends about
Quackit.com! ');" value="Double Click Me!" />
```

To prevent the load from refreshing, you could use the JavaScript `void()` function and pass a parameter of 0 (zero).

Example of void(0):

We have a link that should only do something (i.e. display a message) upon two clicks (i.e. a double click). If you click once, nothing should happen. We can specify the double click code by using JavaScript's "ondblclick" method. To prevent the page reloading upon a single click, we can use "JavaScript:void(0);" within the anchor link.

Code:

```
<a href="JavaScript:void(0);"
ondblclick="alert('Well done! ')">Double Click
Me!</a>
```

Result:

Double Click Me!

Same Example, but without void(0):

Look at what would happen if we didn't use "JavaScript:void(0);" within the anchor link...

Code:

```
<a href="" ondblclick="alert('Well
done! ')">Double Click Me!</a>
```

Result:

Double Click Me!

Did you notice the page refresh as soon you clicked the link. Even if you double clicked and triggered the "ondblclick" event, the page still reloads!

Note: Depending on your browser, your browser might have redirected you to the "/javascript/tutorial/" index page. Either way, JavaScript's "void()" method will prevent this from happening.

Void(0) can become useful when you need to call another function that may have otherwise resulted in an unwanted page refresh.

Refresh code In JavaScript, you refresh the page using `location.reload`.

This code can be called automatically upon an event or simply when the user clicks on a link.

Example JavaScript Refresh code

Typing this code:

```
<!-- Codes by Quackit.com -->
<a href="javascript:location.reload(true)">Refresh this page</a>
```

You can use JavaScript to automatically open print dialogue box so that users can print the page. Although most users know they can do something like "File > Print", a "Print this page" option can be nice, and may even encourage users to print the page. Also, this code can be triggered automatically upon loading a printer friendly version.

Creating a "Print this page"

The following code creates a hyperlink and uses the Javascript print function to print the current page:

```
<a href="JavaScript:window.print();">Print this
page</a>
```

Open a new window in javascript

One of the more common requests I see is how to open a new Javascript window with a certain feature/argument. The Window.open method has been available since Netscape 2.0 (Javascript 1.0), but additional window decoration features have been added to the syntax since then.

Syntax

window.open([URL], [Window Name], [Feature List], [Replace]);

where:

[URL] **Optional**. Specifies the URL of the new window. If no URL is given, the window opens with no page loaded.

[Window Name] **Optional**. Specifies a name for the window. This name can be used to reference the window as a destination for a hyperlink using the HTML TARGET attribute.

[Features] **Optional**. Comma separated strings specifying the features added to the new window. Typical values are booleans (TRUE/FALSE.) Syntax is of the form

[feature] "=" [value] ("," [feature] "=" [value])*


[Replace] **Optional**. Boolean value specifying whether the new window replaces the current window in the browser's history.

```
window.open('dummydoc.htm', 'height=480,width=640');
```

write to new window

How do I write script-generated content to another window?

To write script-generated content to another window, use the method `winRef.document.write()` or `winRef.document.writeln()`, where `winRef` is the [window reference](#), as returned by the `window.open()` method.

To make sure that your script's output actually shows up in the other window, use `winRef.document.close()` after writing the content. As an example, consider the following function that opens a new window with the title **Console** and writes the specified content to the new window. 

In the above example, you might notice that after you write something to the console *several times*, the console window will allow you to navigate back and forth in the output's history. This is not always a desired feature. If you would like to output the new content without creating a new *history entry*, add the following operator after opening the window (and before the first write):

```
docRef = top.winRef.document.open("text/html","replace");
```

Here `winRef` is the window reference returned by the `window.open()` method, and `docRef` is a global variable in which the script stores the reference to your new document.

```
<HTML>
<HEAD>
<TITLE>Writing to Subwindow</TITLE>
<SCRIPT LANGUAGE="JavaScript">
var newWindow
function makeNewWindow() {
    newWindow = window.open("", "", "status,height=200,width=300")
}
function subWrite() {
    if (newWindow.closed) {
        makeNewWindow()
    }
    newWindow.focus()
    var newContent = "<HTML><HEAD><TITLE>A New Doc</TITLE></HEAD>"
    newContent += "<BODY BGCOLOR='coral'><H1>This document is brand new.</
H1>"
    newContent += "</BODY></HTML>"
    newWindow.document.write(newContent)
    newWindow.document.close() // close layout stream
}
</SCRIPT>
</HEAD>
<BODY onLoad="makeNewWindow()">
```

```
<FORM>
<INPUT TYPE="button" VALUE="Write to Subwindow" onClick="subWrite()">
</FORM>
</BODY>
</HTML>
```

Moving and resizing windows

When you have created a window, you can use Javascript to move it or resize it.

Moving windows

A window object has two methods for moving it: `moveTo` and `moveBy`. Both take two numbers as arguments.

The first function moves the window to the specified coordinates, measured in pixels from the top left corner of the screen. The first argument is the horizontal, or X, coordinate, and the second is the vertical, or Y, coordinate.

The second method changes the current coordinates by the given amounts, which can be negative to move the window left or up.

Not all browsers allow you to move a window. A browser with a tabbed interface has one window in each tab, and cannot move or resize them individually. A browser with MDI (e.g., Opera) can only move the windows inside the parent application's window, and only if the document window isn't maximized.

```
window . moveBy(-10,20);
```

Resizing windows

Similar to the methods for moving, there are two methods for resizing a window: `resizeTo` and `resizeBy`. Like for moving, they either set the size absolutely or modify the size relatively to the current size.

Most browsers' standard security setting will not let you resize a window to less than 100 pixels in any direction.

```
window.resizeBy(100,-100);
```

Closing windows

Closing a window is simple when it works. All windows have a `close` method, so you can attempt to close any window you have a reference to.

```
myWindow.close();
```

However, not all windows can be closed without asking the user for permission. This is to protect the user from accidentally losing his browsing history.

The cases where you can close a window without a warning are:

- If a window was opened using Javascript, then it can be closed again without a warning.
- In some browsers, if the browser history contains only the current page, then the window can be closed without warning.
- In some browsers, setting the "window.opener" property to any window object will make the browser believe that the window was opened with Javascript.

Checking whether window has been closed

Given a reference to a window, it is possible to see whether the window has been closed.

```
if (myWindow.closed) { /* do something, e.g., open it again */ }
```

Browsers will generally not allow access to another window's properties, if it contains a page from a different domain. In some browsers, this even includes the `closed` property.

Problems with opening windows

Opening windows is not as safe as it used to be. Pages opening unwanted windows with advertisements have made people distrust new windows. Many users have installed software that prevents unwanted windows from opening.

Such *popup blockers* work in many different ways, which makes it hard to say anything definite. Some examples:

- Proximitron, a rewriting web proxy, can add Javascript that changes the `window.open` function in different ways. One option is to have it always return the current window instead of a new one.
- Some popup blockers work at the operating system level, and automatically close all new windows created by a browser. The call to `window.open` succeeds, but the window disappears before the user can see it.
- Modern browsers have popup-blockers built in (e.g., Mozilla, Opera, or MyIE2). They can be set to disable all new windows, or to allow only those that result from a user action (clicking on a link or button). A blocked popup can give a Javascript security error, stopping the script.
- Some specialized browsers, e.g., PDA's and mobile phones, completely lack the ability to open new windows, and might have no `window.open` function at all. Another specialized browser is WebTV, which can only do full screen pages, and it ignores new windows with sizes below 400 by 300 pixels and new windows with no size specified.

In the face of such diverse and unpredictable opposition, anybody trying to open a window should be prepared for it to fail. And it can fail silently, visibly, or irrecoverably. Some are of the opinion that `window.open` is no longer safe enough to use for anything important.

Windows status

- `window.status = "This message will display in the window status bar."`

The window status bar will not always work when written to immediately when the page loads. Normally it is used when an event happens such as when the user clicks on something or moves the mouse over an item or area on the page. The following code will display a status bar message when the mouse is moved over the home button

- blur() - This function will take the focus away from the window. Calling it as follows will perform the function.
 - `<script language="JavaScript">`
 - `blur()`
 - `</script>`
- close() - This function will close the current window or the named window. Normally a form button is provided to allow the user to close the window and the function is called by the onClick action. Otherwise some other event may be used to close the window.

`window.close()`

- focus() - This function will give the focus to the window.
- moveBy(x,y) - The window is moved the specified number of pixels in the X and Y direction.
- moveTo(x,y) - The window is moved to the specified X and Y pixel location in the browser.
- open("URLname","Windowname",["options"]) - A new window is opened with the name specified by the second parameter. The document specified by the first parameter is loaded. The options are not required, and may be set to values of yes or no as in the below example. Many options are set to default values depending on whether others are set. For example, if the width and height are specified, the resizable option is set as default to no. Also if the toolbar is off, the menubar is set to default of no. The options are:
 - alwaysRaised - If yes, the created window is raised.
 - directories - The value of yes or no determines if the window shows directory buttons or not.
 - height - Specifies the window height in pixels. This is set to an integer value, rather than yes or no.
 - left - Specifies the distance in pixels the new window will be placed from the left side of the screen. This is set to an integer value, rather than yes or no.
 - location - The value of yes or no determines if the window has a location displayed on the address line (or has an address line) or not.

- menubar - The value of yes or no determines if the window has a menubar or not.
- outerWidth - Specifies the outer window width in pixels. This is set to an integer value, rather than yes or no.
- outerHeight - Specifies the outer window height in pixels. This is set to an integer value, rather than yes or no.
- resizable - The value of yes or no determines if the window can be resized by the user or not
- status - The value of yes or no determines if the window has a status bar or not.
- scrollbars - The value of yes or no determines if the window has scroll bars or not.
- toolbar - The value of yes or no determines if the window has a toolbar or not.
- top - Specifies the distance in pixels the new window will be placed from the top of the screen. This is set to an integer value, rather than yes or no.
- width - Specifies the window width in pixels. This is set to an integer value, rather than yes or no.
- z-lock - If yes, the created window is in the background.

Example:

```
<script language="JavaScript">  
open("javahere.html","test",  
"toolbar=no,menubar=no,width=200,height=200,resizab  
le=yes")  
</script>
```

•



Active Server Pages

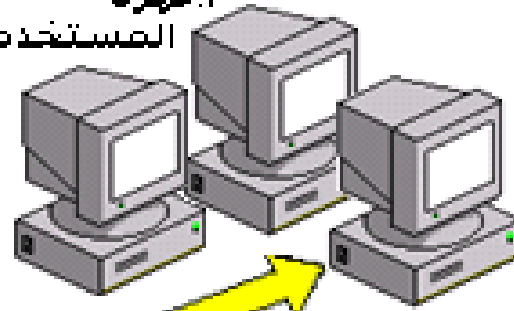


Introduction to the ASP

- "ASP is a technology developed by Microsoft . to create pages and Web applications are strong and dynamic. To create these pages you can add HTML or a scripting languages Scripting language such as VBScript or JavaScript, and you can also connect these pages to a database such as Access or SQL Server".
- ASP is abbreviated to (Active Server Pages)
- using ASP programming language web pages to do some events code.
- ASP file code implemented on the server side (Server Side), either the result be sent to the user's computer to display.



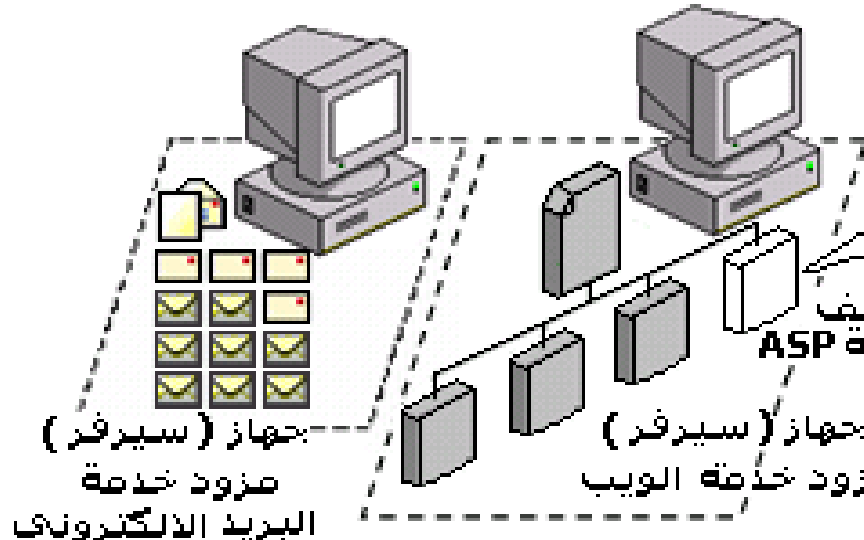
اجهزة
المستخدمين



جانب الزبون
(المستخدم)



جانب مزود الخدمة
(السيرفر)



تفخذ شجرة ملف صفحة
ASP على جانب السيرفر
و ترسل النتيجة لتعرض
على المستخدم جهاز



ASP formats & Basic concepts

- You can not see ASP code

By displaying the source code in the browser, you are the only one who can repeat to see ASP code when your design for active server pages, because script implemented on the server before you show the results sent to the browser.

Files ASP naturally contain medals of HTML, and also contain script server, and the texts of the ASP writes always between those delineated by `<% %>`, and script server implemented on the server, and contains Script server on equations, and sentences, and transactions correct languages script that you use



The difference between the ASP and HTML?

- 1 - When a user requests the HTML file, the server sends the same HTML file to display the matching program.
- 2 - When a user requests the ASP file, IIS passes the request to the ASP engine, the ASP engine reads the ASP file line by line, and executes the script inside the file, in the end, the ASP file is due to the browser as an HTML file



- ASP pages different from HTML pages that address these pages are from the side of the device server not the user's computer, where it is installed in a server program features a dynamic link library called ASP.DLL, when a user requests a page extension asp service provider processes the orders existing ASP this and sends the result, and this result is a HTML commands to display in the browser on the user's computer. This method provides a degree of safety to these pages as they preserve the rights of programmed versions, where the user can not see ASP commands when viewing the page source, but sees HTML commands a result of the treatment. The question arises, do we need a host (server) to tackle the ASP pages that we create?! Of course, not! All we need is a web server programs that address these pages, and the inauguration of this program we have made our server system and a user at the same time.

Web server software:

There are two suitable:

1 - PWS program, a shortcut Personal Web Server

2 - Program IIS which is an a shortcut for Internet Information Server

The choice of one of these programs on the operating system you have





Active Server Pages

IIS - Internet Information Server

- IIS is a set of Internet-based services for servers created by Microsoft for use with Microsoft Windows.
- IIS comes with Windows 2000, XP, Vista, and Windows 7. It is also available for Windows NT.
- IIS is easy to install and ideal for developing and testing web applications.

PWS - Personal Web Server

- PWS is for older Windows system like Windows 95, 98

How to Install IIS on Windows 7 and Windows Vista

Follow these steps to install IIS:

- Open the Control Panel from the Start menu
- Double-click Programs and Features
- Click "Turn Windows features on or off" (a link to the left)
- Select the check box for Internet Information Services (IIS), and click OK


How to Install IIS on Windows XP and Windows 2000

Follow these steps to install IIS:

- On the Start menu, click Settings and select Control Panel
- Double-click Add or Remove Programs
- Click Add/Remove Windows Components
- Click Internet Information Services (IIS)
- Click Details
- Select the check box for World Wide Web Service, and click OK
- In Windows Component selection, click Next to install IIS

After you have installed IIS or PWS follow these steps

- Look for a new folder called **Inetpub** on your hard drive
- Open the Inetpub folder, and find a folder named **wwwroot**
- Create a new folder, like "MyWeb", under wwwroot
- Write some ASP code and save the file as "test1.asp" in the new folder
- Make sure your Web server is running (see below)
- Open your browser
"http://localhost/MyWeb/test1.asp", to view your first web page

- 
- An ASP file normally contains HTML tags, just like an HTML file. However, an ASP file can also contain server scripts, surrounded by the delimiters `<%` and `%>`.
 - Server scripts are executed on the server, and can contain any expressions, statements, procedures, or operators valid for the scripting language you prefer to use.

Script languages used with ASP:

There are two different types of (Script Languages) can be used to write the active code on the ASP pages are:

1 - VBScript (the default language permitted use in the pages of the ASP).

2 - JavaScript.

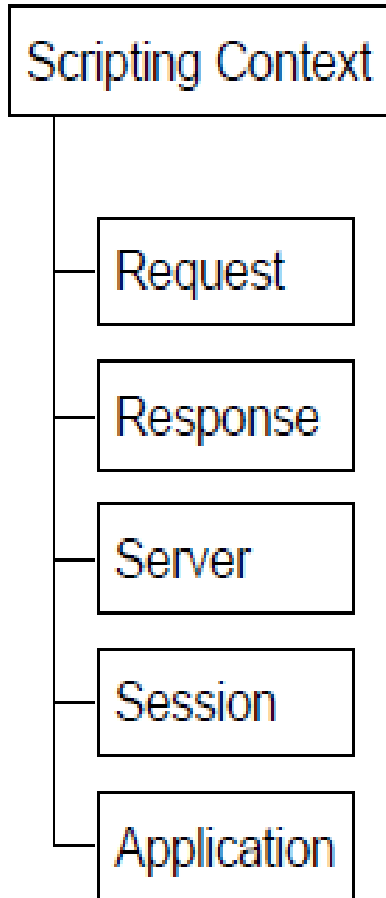
VBscript

```
<%@ language= "VBscript" %>
```

Javascript

```
<%@ language= "Javascript" %>
```


Overview



Request: To get information from the user

Response: To send information to the user

Server: To control the Internet Information Server

Session: To store information about and change settings for the user's current Web-server session

Application: To share application-level information and control settings for the lifetime of the application

Response object

- used the Response object to send information / results from a server to the user's computer
- Response Object , number of properties (Attributes) and approach (Methods) are as follows
- ```
<%@ language= "VBscript" %>
<html dir="rtl">
<body>
 <%
 response.write ("مرحباً بكم في برمجة مواقع الويب")
 %>
</body>
</html>
```

- Definition of variables:  
Variables are used to store data. The following example will explain how variables are defined in the ASP

- ```
<html dir="rtl">
<head>
</head>
<body>
  <%
    dim name
    name=" برمجة مواقع الويب "
    response.write("<center><B> مرحباً بكم في " &name&
"</b</center>")
  %>
</body>
</html>
```

- dim : used to define the variables.
name: the name of the variable, and we set a value stored inside.
In print command notes that we did not put the variable in quotation marks "" because we want to print the value of this variable and not print the same variable.
When we want to print the value of the variable, and we want to include HTML elements with him must be separated between the two signal & As is shown in the top

```
<%
```

```
dim fname(5),i
  fname(0)="مرحبا"
  fname(1)="بكم"
  fname(2)="في"
  fname(3)="برمجة"
  fname(4)="مواقع الويب"
  for i=0 to 4
    response.write("<center><b>" &fname(i)& "</b><center>")
  next
%>
```



Active Server Pages

- **What is localhost**

Let us first see, what we mean by a hostname.

Whenever you connect to a remote computer using its URL, you are in effect calling it by its hostname.

For example, when you type in `http://www.google.com/`

you are really asking the network to connect to a computer named

`www.google.com`. It is called the “hostname” of that computer.

`localhost` is a special hostname. It always references your own machine. So what you just did, was to try to access a webpage on your own machine (which is what you wanted to do anyway.) For testing all your pages, you will need to use `localhost` as the hostname.

- **Response**
- The Response Object is responsible for sending data from the client to the server.

يملك الكائن Response Object عدد من الخصائص (Attributes) و المنهج (Methods) وهي كالتالي:-

منهج الكائن Response Object

الوصف	المنهج
تضيف عنوان جديد للـ HTTP وقيمة جديدة.	AddHeader
تقوم بمسح جميع البيانات المخزنة داخل الـ Buffer	Clear
توقف عملية المعالجة للسكربتات المخزنة داخل الـ buffer، وتظهر الناتج الحالي.	End
يرسل المحتوي المخزن داخل الـ buffer إلى برنامج التصفح.	Flush
ترجع المستخدم مباشرة إلى رابط آخر.	Redirect

Request object

used to deliver data sent by the user's computer to a server through HTTP request.

- **Required object**

- There are two ways to get form information:

- 1-The Request.QueryString command

- 2- The Request.Form command.

- **1- Request.QueryString**

The Request.QueryString command collects the values in a form as text. Information sent from a form with the GET method is visible to everybody (in the address field). Remember that the GET method limits the amount of information to send.

- Request.QueryString
Method="get"
action = "filename.asp"
- **Syntax**
- **Request.QueryString(variable)[(index)|.Count]**

- **2-Request.Form**
- Command Request.form, which is not much different from something Request.querystring, it is used to gather the values of the form with the use of method Post, which may not be visible to anyone, and it does not specify the amount of.
- **Syntax**
- **Request.Form(element)[(index)|.Count]**
- Request.Form
Method="post"
action = "filename.asp"

GET and POST

- One thing we ignored in our discussion about forms was that the METHOD by which the form is submitted may be one of the two: GET or POST.
- **When to use GET?**
- Any data that you pass via a GET can be retrieved in your script by using the Request.QueryString collection. GET may be used for small amounts of data
- – the reason being that, the data items are appended to the URL by your
- browser, and obviously, you cannot have an infinitely long URL (with the
- QueryString).
- **When to use POST?**
- Almost always. Stick to POST for your forms, and be sure to use the Request.Form collection to access them (and *not* the Request.QueryString
- collection.)

- Simpleform.asp

- `<html>`

- `<body>`

- `<%`

- `request.querystring("fname")`

- `request.querystring("lname")`

- `%>`

- `<form method="get" action="simpleform.asp">`

- `fname : <input type="text" name="fname">`

- `</br >`

- `lname: <input type="text" name="lname">`

- `</br ></br >`

- `<input type="submit" value="sent data" >`

- `</form>`

- `</body>`

- `</html>`

- Run

- <http://localhost/simpleform.asp?fname=shatha &lname=Habeeb>

- ```
<html dir="rtl">
<body>
 <%
 request.form("username")
 request.form("pass")
 %>
 <form method="post" action="form2.asp">
 اسم المستخدم: <input type="text" name="fname"></br >
 كلمة المرور: <input type="text" name="lname">
 </br ></br >
 <input type="submit" value="إرسال البيانات">
 </form>
 <% ="اسم المستخدم:" &request.form("username") %></br>
 <% ="كلمة المرور:" &request.form("pass") %>
</body>
</html>
```

## Form1.asp

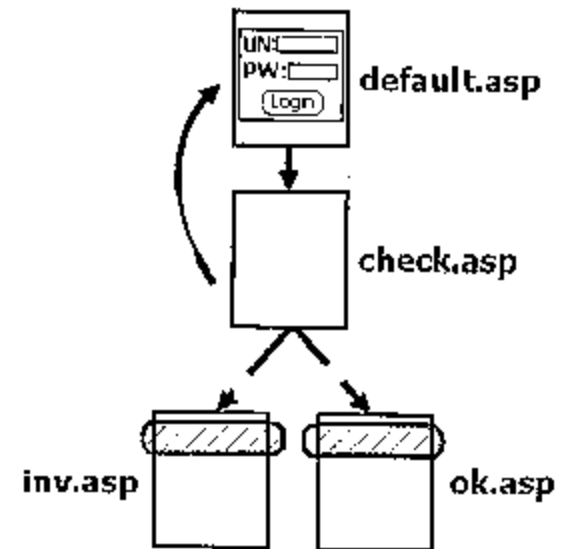
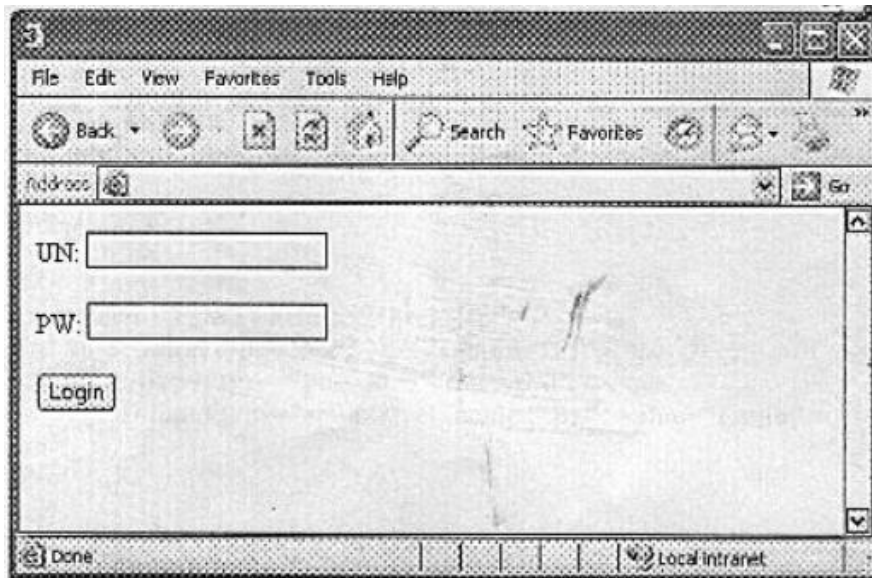
```
<html dir="rtl">
 <body>
 <form method="get" action="form2.asp">
 fname: <input type="text" name="fname">
 </br >
 lname: <input type="text" name="lname">
 </br ></br >
 <input type="submit" value="send data">
 </form>
 </body>
</html>
```

## Form2.asp

```
<html dir="rtl">
 <body>
 <%
 dim fname,lname
 request.querystring("fname")
 request.querystring("lname")
 %>
 fname:<%= ""&fname""%></br>lname:<%= ""&lname
 &""%>
 </body>
</html>
```

# Login Application:

Typically, "Login Application" is used to allow the authorized users to access private and secured areas, by using User-Name and Password strings.



```
<head>
</head>
<body>
<form method="POST" action="check.asp">
 UN: <input type="text" name="T1"> <p>
 PW: <input type="password" name="T2"> <p>
 <input type="submit" name="B1" value="Login" >
</form>
</body>
</html>
```

الصفحة  
default.asp

```
<html>
<head>
</head>
<body>
<%
 un=request.form("T1")
 pw=request.form("T2")
 if un="" OR pw="" then
 response.redirect("default.asp")
 else
 if un=" sha AND pw="123" then
 respo direct("ok.asp")
 else
 response.redirect("inv.asp")
 end if
 end if
%>
</body>
</html>
```

الصفحة  
check.asp

OR

```
<html>
<head>
</head>
<body>
 Welcome
</body>
</html>
```

الصفحة  
ok.asp

```
<html>
<head>
</head>
<body>
 Invalid UN or PW
</body>
</html>
```

الصفحة  
inv.asp



- However, the internal pages ('ok. asp' and 'in .asp' pages) must be having "**Guarding Code**" to prevent any direct access to them. So if any user write the following address (http:// ..... / ok. asp) to access the 'ok. Asp' page without coming from login-form (in other word to by- passing the checking code), the guarding-code must be prevent this unauthorized access and the process must be return to the 'default, asp' page The "Guarding Code" must be written inside all internal pages

- The Guarding-Code is work by check if the user access to this page by follows authorized-path or not. If the user access 'default.asp' page and enter correct username and password strings, then the 'check.asp' page must be not only redirect the process to the 'ok.asp' page but must register the username inside cookie file. So when access the 'ok.asp' page the process must be check cookie file, if cookie hold username then this means that the user access 'ok.asp' page by follow the authorized path, else if cookie hold nothing then this means that the user try to access to the 'ok.asp' page by by-passing checking code.

# Cookie

- A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With ASP, you can both create and retrieve cookie values.

# How to Create a Cookie?

- The "Response.Cookies" command is used to create cookies.
- <%  
Response.Cookies("firstname")="Alex"
- %>
- It is also possible to assign properties to a cookie, like setting a date when the cookie should expire:
- <%  
Response.Cookies("firstname")="Alex"  
Response.Cookies("firstname").Expires=#May 10,2012#  
%>

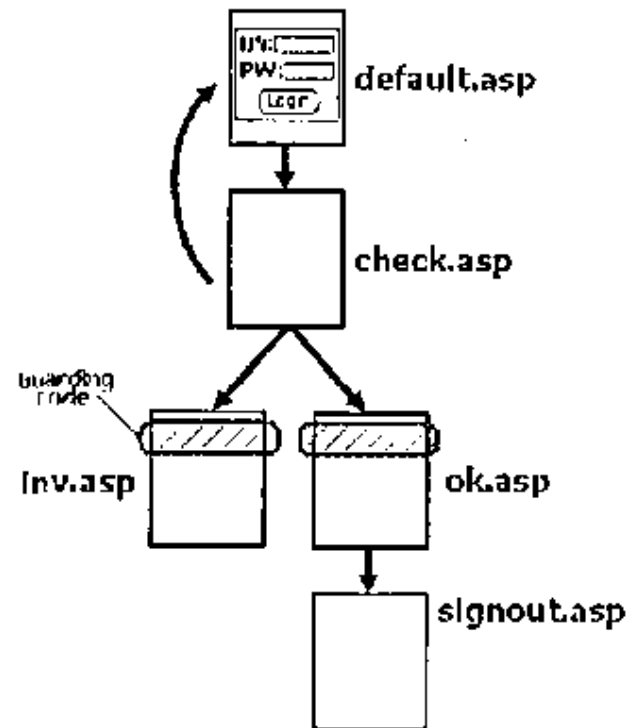
# How to Retrieve a Cookie Value?

- The "Request.Cookies" command is used to retrieve a cookie value.
- ```
<%  
fname=Request.Cookies("firstname")  
response.write("Firstname=" & fname)  
%>
```
- **Output:** Firstname=Alex

A Cookie with Keys

- If a cookie contains a collection of multiple values, we say that the cookie has Keys.
- In the example below, we will create a cookie collection named "user". The "user" cookie has Keys that contains information about a user:
- ```
<%
Response.Cookies("user")("firstname")="John"
Response.Cookies("user")("lastname")="Smith"
Response.Cookies("user")("country")="Norway"
Response.Cookies("user")("age")="25"
%>
```

- However, when user wants to leave 'ok.asp' page, so the user must be sign out this page, this means that the user must put nothing inside cookie file.



# Read all Cookies

Look at the following code:

- ```
<%  
Response.Cookies("firstname")="Alex"  
Response.Cookies("user")("firstname")="John"  
Response.Cookies("user")("lastname")="Smith"  
Response.Cookies("user")("country")="Norway"  
Response.Cookies("user")("age")="25"  
%>
```
- Assume that your server has sent all the cookies above to a user

Active Server Pages: Open, Read and Create files



FileSystemObject

- The FileSystemObject object is used to access the file system on a server. This object can manipulate files, folders, and directory paths. It is also possible to retrieve file system information with this object.

Syntax

- **Scripting.FileSystemObject**
- Set fs = CreateObject("Scripting.FileSystemObject")
- Set a = fs.CreateTextFile("c:\testfile.txt", True)
- a.WriteLine("This is a test.")
- a.Close



I- Open and Read content from a text file

```
<%
```

```
Set fs = CreateObject("Scripting.FileSystemObject")
```

```
Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
```

```
filecontent = wfile.ReadAll
```

```
wfile.close
```

```
Set wfile=nothing
```

```
Set fs=nothing
```

```
response.write(filecontent)
```

```
%>
```



ReadLine

```
<%  
    Set fs = CreateObject("Scripting.FileSystemObject")  
    Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")  
    firstname = wfile.ReadLine  
    lastname = wfile.ReadLine  
    theage = wfile.ReadLine  
wfile.close  
    Set wfile=nothing  
    Set fs=nothing  
%>  
Your first name is <% =firstname %><BR>  
Your last name is <% =lastname %><BR>  
Your are <% =thaage %> years old<BR>
```



AtEndOfStream

- <%
Set fs = CreateObject("Scripting.FileSystemObject")
- Set wfile = fs.OpenTextFile("c:\Mydir\myfile.txt")
- counter=0
do while not wfile.AtEndOfStream
 counter=counter+1
 singleline=wfile.readline
 response.write (counter & singleline & "
")
loop
- wfile.close
Set wfile=nothing
Set fs=nothing
- %>



2- Create and Write a text file

Example 1: The basic code we need to create a file is very similar to that one we have used to open a file:

```
<%  
dim fs,f  
set fs=Server.CreateObject("Scripting.FileSystemObject")  
set f=fs.CreateTextFile("c:\test.txt",true)  
f.WriteLine("Hello World!")  
f.Close  
set f=nothing  
set fs=nothing  
%>
```



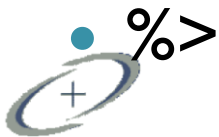
The #include Directive

- You can insert the content of one ASP file into another ASP file before the server executes it, with the #include directive.
- The #include directive is used to create functions, headers, footers, or elements that will be reused on multiple pages.



Here is a file called "mypage.asp"

- `<html>`
`<body>`
`<h3>Words :</h3>`
`<p><!--#include file="inclu.inc"--></p>`
`<h3>The time is:</h3>`
`<p><!--#include file="time.inc"--></p>`
`</body>`
`</html>`
- `<%`
- `response.write(date())`
- `%>`



What is ADO?

ADO can be used to access databases from your web pages

- ADO is a Microsoft technology
- ADO stands for **A**ctiveX **D**ata **O**bjects
- ADO is automatically installed with Microsoft IIS
- ADO is a programming interface to access data in a database



Accessing a Database from an ASP Page

The common way to access a database from inside an ASP page is:

- Create an ADO connection to a database
- Open the database connection
- Create an ADO recordset
- Open the recordset
- Extract the data you need from the recordset
- Close the recordset
- Close the connection

Before a database can be accessed from a web page, a database connection has to be established.



- The ADO Connection Object is used to create an open connection to a data source. Through this connection, you can access and manipulate a database.
- If you want to access a database multiple times, you should establish a connection using the Connection object. You can also make a connection to a database by passing a connection string via a Command or Recordset object. However, this type of connection is only good for one specific, single query.

set objConnection

=Server.CreateObject("ADODB.connection")



Methods

- Cancel
- Cancels an execution
- Close
- Closes a connection
- Execute
- Executes a query, statement, procedure or provider specific text
- Open
- Opens a connection
- OpenSchema
- Returns schema information from the provider about the data source



Create a DSN-less Database Connection

- The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft Access database on your web site.
- If you have a database called "northwind.mdb" located in a web directory like "c:/webdata/", you can connect to the database with the following ASP code:
- ```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/webdata/northwind.mdb"
%>
```



- Note, from the example above, that you have to specify the Microsoft Access database driver (Provider) and the physical path to the database on your computer.

## **ADO Recordset**

- The ADO Recordset object is used to hold a set of records from a database table. A Recordset object consist of records and columns (fields).
- To be able to read database data, the data must first be loaded into a recordset.
- Suppose we have a database named "Northwind", we can get access to the "Customers" table inside the database with the following lines:



set

```
objRecordset=Server.CreateObject("ADODB.recordset")
```

- <%

```
set
```

```
conn=Server.CreateObject("ADODB.Connection")
```

```
conn.Provider="Microsoft.Jet.OLEDB.4.0"
```

```
conn.Open "c:/webdata/northwind.mdb"
```

```
set
```

```
rs=Server.CreateObject("ADODB.recordset")
```

```
rs.Open "Customers", conn
```

```
%>
```

- When you first open a Recordset, the current record pointer will point to the first record and the BOF and EOF properties are False. If there are no records, the BOF and EOF property are True.



## Create an ADO SQL Recordset

We can also get access to the data in the "Customers" table using SQL:

- ```
<%  
set conn =  
Server.CreateObject("ADODB.Connection")  
conn.Provider="Microsoft.Jet.OLEDB.4.0"  
conn.Open "c:/webdata/northwind.mdb"  
set rs=Server.CreateObject("ADODB.recordset")  
rs.Open "Select * from Customers", conn  
%>
```



Structured Query Language (SQL)

- The Structured Query Language (SQL) forms the backbone of all relational databases. This language offers a flexible interface for databases of all shapes and sizes and is used as the basis for all user and administrator interactions with the database.



Definition of a Relational Database

- A relational database is a collection of relations or two-dimensional tables.

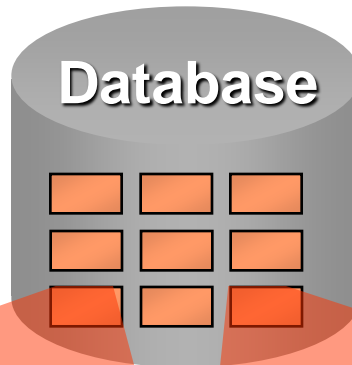


Table Name: **EMP**

| EMPNO | ENAME | JOB | DEPTNO |
|-------|-------|-----------|--------|
| 7839 | KING | PRESIDENT | 10 |
| 7698 | BLAKE | MANAGER | 30 |
| 7782 | CLARK | MANAGER | 10 |
| 7566 | JONES | MANAGER | 20 |

Table Name: **DEPT**

| DEPTNO | DNAME | LOC |
|--------|------------|----------|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |



SQL statement
is entered

```
SQL> SELECT loc  
      FROM dept;
```

Statement is sent to
database

Database



Data is displayed

```
LOC  
-----  
NEW YORK  
DALLAS  
CHICAGO  
BOSTON
```



Basic SELECT Statement

```
SELECT    [DISTINCT] {*, column [alias], ...}  
FROM      table;
```

- ◉ SELECT identifies *what* columns.
- ◉ FROM identifies *which* table.



Selecting All Columns

```
SQL> SELECT *  
        FROM dept;
```

| DEPTNO | DNAME | LOC |
|--------|------------|----------|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |



Selecting Specific Columns

```
SQL> SELECT deptno, loc  
FROM dept;
```

| DEPTNO | LOC |
|--------|----------|
| 10 | NEW YORK |
| 20 | DALLAS |
| 30 | CHICAGO |
| 40 | BOSTON |



Arithmetic Expressions

Create expressions on NUMBER and DATE data by using arithmetic operators. •

| Operator | Description |
|----------|-------------|
| + | Add |
| - | Subtract |
| * | Multiply |
| / | Divide |



Using Arithmetic Operators

```
SQL> SELECT ename, sal, sal+300  
emp;      FROM
```

| ENAME | SAL | SAL+300 |
|--------|------|---------|
| KING | 5000 | 5300 |
| BLAKE | 2850 | 3150 |
| CLARK | 2450 | 2750 |
| JONES | 2975 | 3275 |
| MARTIN | 1250 | 1550 |
| ALLEN | 1600 | 1900 |

...

14 rows selected.



Limiting Rows Selected

Restrict the rows returned by using the  WHERE clause.

```
SELECT          [DISTINCT] { * | column [alias], ... }  
FROM            table  
[WHERE          condition(s) ] ;
```

The WHERE clause follows the FROM clause. 



Using the WHERE Clause

```
SQL> SELECT ename, job, deptno  
       FROM emp  
       WHERE job='CLERK' ;
```

| ENAME | JOB | DEPTNO |
|--------|-------|--------|
| JAMES | CLERK | 30 |
| SMITH | CLERK | 20 |
| ADAMS | CLERK | 20 |
| MILLER | CLERK | 10 |



Comparison Operators

| Operator | Meaning |
|----------|--------------------------|
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| < | Less than |
| <= | Less than or equal to |
| <> | Not equal to |



Using the Comparison Operators

```
SQL> SELECT ename, sal, comm  
       FROM emp  
       WHERE sal<=comm;
```

| ENAME | SAL | COMM |
|--------|------|------|
| MARTIN | 1250 | 1400 |



Other Comparison Operators

| Operator | Meaning |
|------------------------------|---------------------------------------|
| BETWEEN
...AND... | Between two values (inclusive) |
| IN(list) | Match any of a list of values |
| LIKE | Match a character pattern |
| IS NULL | Is a null value |



Using the BETWEEN Operator

Use the BETWEEN operator to display rows based on a range of values.

```
SQL> SELECT  ename, sal
FROM        emp
WHERE       sal BETWEEN 1000 AND 1500;
```

| ENAME | SAL |
|--------|------|
| MARTIN | 1250 |
| TURNER | 1500 |
| WARD | 1250 |
| ADAMS | 1100 |
| MILLER | 1300 |

Lower
limit

Higher
limit



Using the IN Operator

Use the IN operator to test for values in a list.

```
SQL> SELECT empno, ename, sal, mgr
       FROM emp
       WHERE mgr IN (7902, 7566, 7788);
```

| EMPNO | ENAME | SAL | MGR |
|-------|-------|------|------|
| 7902 | FORD | 3000 | 7566 |
| 7369 | SMITH | 800 | 7902 |
| 7788 | SCOTT | 3000 | 7566 |
| 7876 | ADAMS | 1100 | 7788 |



Using the LIKE Operator

- Use the LIKE operator to perform wildcard searches of valid search string values.
- Search conditions can contain either literal characters or numbers.
 - % denotes zero or many characters.
 - _ denotes one character.

```
SQL> SELECT   ename  
        FROM   emp  
        WHERE  ename LIKE 'S%';
```



Using the LIKE Operator

- You can combine pattern-matching characters.

```
SQL> SELECT  ename  
        FROM    emp  
        WHERE   ename LIKE '_A%';
```

ENAME

MARTIN

JAMES

WARD



Using the IS NULL Operator

- Test for null values with the IS NULL operator.

```
SQL> SELECT  ename, mgr  
          FROM    emp  
          WHERE   mgr IS NULL;
```

| ENAME | MGR |
|-------|-------|
| ----- | ----- |
| KING | |



Using the NOT Operator

```
SQL> SELECT ename, job  
       FROM emp  
       WHERE job NOT IN ('CLERK', 'MANAGER', 'ANALYST');
```

| ENAME | JOB |
|--------|-----------|
| ----- | ----- |
| KING | PRESIDENT |
| MARTIN | SALESMAN |
| ALLEN | SALESMAN |
| TURNER | SALESMAN |
| WARD | SALESMAN |



ORDER BY Clause

- Sort rows with the ORDER BY clause
 - ASC: ascending order, default
 - DESC: descending order
- The ORDER BY clause comes last in the SELECT statement.

```
SQL> SELECT      ename, job, deptno, hiredate
      FROM        emp
      ORDER BY    hiredate;
```

| ENAME | JOB | DEPTNO | HIREDATE |
|-------|----------|--------|-----------|
| SMITH | CLERK | 20 | 17-DEC-80 |
| ALLEN | SALESMAN | 30 | 20-FEB-81 |
| ... | | | |

14 rows selected.



The INSERT Statement

- Add new rows to a table by using the INSERT statement.

```
INSERT INTO table [(column [, column...)]]  
VALUES      (value [, value...]);
```

- Only one row is inserted at a time with this syntax.



Inserting New Rows

- Insert a new row containing values for each column.
- List values in the default order of the columns in the table.
- Optionally list the columns in the INSERT clause.

```
SQL> INSERT INTO      dept (deptno, dname, loc)
      VALUES          (50, 'DEVELOPMENT', 'DETROIT');
1 row created.
```

- Enclose character and date values within single quotation marks.



The UPDATE Statement

- Modify existing rows with the UPDATE statement.

```
UPDATE      table  
SET         column = value [, column = value, ...]  
[WHERE     condition];
```

- Update more than one row at a time, if required.



Updating Rows in a Table

- Specific row or rows are modified when you specify the WHERE clause.

```
SQL> UPDATE emp
      SET deptno = 20
      WHERE empno = 7782;
```

1 row updated.

- All rows in the table are modified if you omit the WHERE clause.

```
SQL> UPDATE employee
      SET deptno = 20;
```

14 rows updated.



The DELETE Statement

- You can remove existing rows from a table by using the DELETE statement.

```
DELETE [FROM] table  
[WHERE condition];
```



Deleting Rows from a Table

- Specific rows are deleted when you specify the WHERE clause.

```
SQL> DELETE FROM      department
        WHERE          dname = 'DEVELOPMENT';
1 row deleted.
```

- All rows in the table are deleted if you omit the WHERE clause.

```
SQL> DELETE FROM      department;
4 rows deleted.
```

