LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #1**

**JANUARY 8, 2001**

PIC 40
Chris Pollett
http://www.math.ucla.edu/~cpollett

Read: Castro App. B & D


creates dynamically
a webpage to send back

Perl script may create whole web page or maybe only add some lines to an existing page (latter called server side incl.)

Another setup
On server have one document

| |
|---|
| HTML |
| Script |
| HTML |

← some commonly used scripting languages (server side scripting)
PHP (based on Perl)
ASP (Visual Basics)

Overview:
Will also talk about accessing databases from Perl
Last topic is JavaScript which helps add functionality to webpages on the client side. e.g. if user's mouse or arrow goes past certain part of screen; changes icon

e.g. Also verifies that an application form is filled out correctly by client before sending it to the server

UNIX & DOS
To get to your Unix account:
either selecting run command under start button then type telnet
pic.ucla.edu (then asked for login)
or under your browser in the URL line
telnet://pic.ucla.edu
then it'll ask for your log-in & password

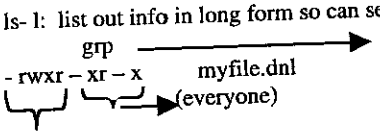
To launch DOS
select it from the start menu.

Useful UNIX/DOS commands:
Look @ directory you're currently in

| UNIX | DOS |
|---|---|
| ls | dir |

UNIX & DOS have option flags you can set
e.g.
UNIX (ls-a)
shows all files including those beg. w/ a prd.

ls- l: list out info in long form so can see permission




can read & exec.
but can't write to it which means can't delete it.

user is allowed to read/write/execute the file.
now you know the directories you write into Directory → folder

To change directory
e.g.
cd. ← changes to current directory
cd .. ← moves you up one directory
cd ~ ← goes to home directory
cd bob ← goes to bob subdir (another folder w/in main) of current dir
cd/ ← root directory (desktop)
cd ~user ← goes to useres home dir.
cd/ etc/password ← goes to password subdir of the etc subdir of the root

Commonly used commands:

pwd ← print working direct.
mkdir ← make a new dir
(mk dir bob)
rm (indosdelete) ← deletes a file rm bob
rm- r bob ← deletes bob subdirectory, all of its children.

**END OF LECTURE** **********

**JANUARY 10, 2001**

Talk about networking involved in transmitting HTML Documents

UNIX/DOS commands:

| UNIX | DOS | | |
|---|---|---|---|
| man | help | man ps | returns manual page for ps command |

man -k keyword

e.g. man -k .gif ← list all commands associated with .gif extension

| | | |
|---|---|---|
| mv | rename | move<br>mv a file to some new location |

e.g.
mv bob sally ← has effect of renaming bob to sally

mv bob . . ← moves file bob into parent directory of current one.

ps ← list your current running process

ps- a ← list all processes

ps– uax ← list by people, all processes you'd be interested in.

kill process # ← gets rid of a running process
can find this number by doing ps

e.g. ps
a.out 755 ← output process PID
kill 755 ← stops a.out

kill- 9 PID ← really kills a process that won't die
or kill- TERM PID

To change permissions:
chmod ← sets permissions on a file
e.g. chmod trx file

ug gives everyone permission
user group to read and execute
w ← write perm
chmod u + x file

To take away permissions:
chmod a-rx file

takes away read execute
u g a permission
| | |
chmod 7 5 5 file

111 101 101
rwx rwx rwx

chmod-R 755 dir ~ sets permission on dir and all subdirectories

To run netscape type:
netscape &
means run netscape on the background

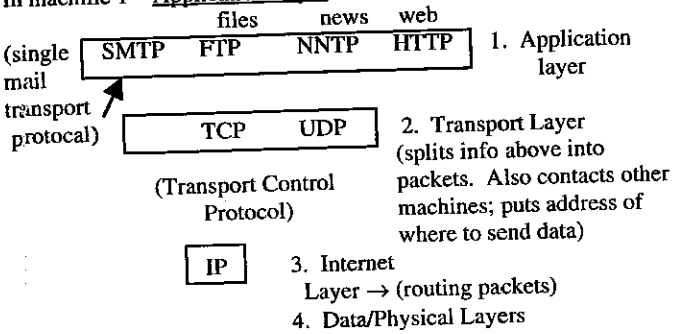
Editors Available

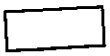
vi
pico - probably easiest pico file
textedit file &
emacs

Networking involved to get HTML document from machine on to your browser

Picture of what Network looks like


In machine 1 – Application Layer





It is here where info transferred to machine 2's Data/Physical Layer

Example of using HTTP protocol to manually get a webpage

telnet www.math.ucla.edu 80
port (listens for incoming messages) where the webserver at the math dept listens HTTP processes handled sep.

connected to arachne.math.ucla.edu
Escape character is '^]'
you type → GET/ index.html
uppercase <HTML>
server resp. <HEAD><TITLE>
w/ document UCLA MATH Dept.
.
.
</HTML>
connection closes

Other HTTP commands

HEAD ~ gets head of HTML doc.
PUT ~ puts a doc on server
POST ~ for forms
DELETE ~ removes from server
LINK
UNLINK

**END OF LECTURE** **********

**JANUARY 12, 2001**

Last Day – talked a little about http & networks

Today – will talk a little about web servers, htaccess and then start talking about HTML docs.

Webservers

| | | |
|---|---|---|
| Apache Webservers | 62% | • percentages of the market |
| Microsoft | 20% | |
| Netscape | 6% | |

Apache Webservers
Apache uses (usually) 3 configuration files.
(can actually have just httpd.conf)
files in:
/etc/httpd/conf

The 3 files are:
1. httpd.conf – controls things like whether webserver under inetd or stand alone. How many child processes are immed. created. What port on the machine to listen to.

2. srm.conf – controls how to map http:/addresses to a particular file on the machine.
   1. e.g. http://www.math.ucla.edu
   → /usr/httpd
   2. http://www.math.ucla.edu/~cpollett
   → /usr/home/cpollett/public_html

3. access.conf – responsible for deciding what kind of document req. are allowed (legal)

If access.conf allows overrides then indiv. users can set .htaccess file in their web directory and modify access.conf specification for that directory

Example .htaccess file

Both .htaccess files & HTML use tags of the form
<TAG> code to be tagged </TAG>

The format of .htaccess files similar to HTML but

```
# This is a comment in .htaccess
# Not a comment in HTML files.

<LIMIT GET> controls which get requests are legal, could
            also control POST, etc.
order allow, deny
deny from    128. 97.4.242
                    deny requests for GET document
allow from all      from this address
otherwise can serve
document
</LIMIT>

XBitHack on
# make it so server side includes work for any
# executable file in this directory
# if SSI or CGI not enabled
# then could uncomment
# Options + Includes + ExecCGI
SSI # Add Handler server – parsed .shtml
CGI # Add Handler cgi-script .cgi, .pl

DirectoryIndex test.html
# default page to serve is
# test.html
<FILES bob.html> ← password protect a file
AuthUser File    /u/willow/h1/fa/cpollett/.htpasswd
AuthType Basic ← password protocol    file where password
AuthName  "Enter Password bob"        stored
require valid-user                    what will print when asking for
</FILES>                              password
# To setup a password
# file can use htpasswd from web

Error Document 404   http://address/error.html
# could handle other errors too
# END .htaccessfile.
```

Example HTML File

```
<HTML> <!- - HTML files begin with <html>
end with </html> BTW this is an HTML
comment - - >

<!- - HTML doc's split into two components a head and a body - =>

    <HEAD>
        <TITLE>
appears in ← my first homepage
title bar of    </TITLE>
the browser

</HEAD>

<BODY       bg color = "white"
            text = "#000000" >
                red  green   blue
<P> First paragraph </P>
<P> 2nd paragraph </P>

</BODY>
</HTML>
```

Netscape  My first Homepage

**END OF LECTURE AND SET #1 ****************************

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #2**

**JANUARY 17, 2001**

Last time talked about
.htaccess
Gave a list HTML document

Today:
**Look at HTML documentss in more detail**

**Recall basic HTML doc looks like**

<html> ← not case sensitive
<head> WRT tags but HTML is, so use lowercase
(↑ with respect to)

</head>
<body> ← stuff printed to screen </body></html>

**I. What markups can appear in the head?**

<title> my page </title>


← what it looks like

Meta Tags

<meta> tags: used to tell the client and server what to do with the documents
↓ talks to server

Ex 1. <meta http – equiv = "Content-Type"
content = "text/html charset = iso – 8859 – 1"/>

tells it what to do — end of tag for XHTML

Tells server to put text/html as mime-type of file and that this is an html file using an English character set

Client uses mime-type to figure out how to draw doc.
This tag is optional

Ex 1:
<meta http-equiv = "refresh"
content = "5; http://my URL" / >
seconds

tells server to instruct client to request the page http://myURL in 5 seconds.

Client Side Tags
often used by search engines when indexing pages
all are optional

Ex 2:
<meta name = "Authors" content = "me" />

client side tag — what to do

<meta name = "description" content = "whatever"
↑
what page is about

<meta name = "keywords" content = "me, myself, I" />
↑
keywords to index from document?

Suppose you don't want your page or any links from your page to be indexed by browser (client) then use:

<meta name = "ROBOTS" content = "NOINDEX, NOFOLLOW"

* *

(optional * tells robots not to index page)
* * tells robots not to follow the links out of this page.

Last thing which may appear in header is stylesheet

<link rel = "stylesheet" href = "test.css" type = "text/css"/>

this says, test.css is the style sheet of the current page and it is of type css2.

style sheet: allows the writer to override the default behaviors of the tags which appear in the body of your document.

Ex 1: test.css

/* this is a stylesheet comment */

p {text-align: right; color: blue}

the p says paragraph should be aligned to right and printed in blue

h1 {text-align: center}
↑
large headlines

II. Now let's look at body of html file
<body>

<body>
**HEADLINES –**
<h1> This is a large headline </h1>
(headlines print in bigger than normal font)
range from h1-h6

 This is a paragraph

 This is another

So far document looks like this:

This is a headline
This is a paragraph
This is another

IMAGES - Including images

<img src = "URL" />

(embed this in your doc – can be in a paragraph)

LINKS – Including links

this is what's underlined

<a href = "URL"> text </a>

PRINT: changing print styles

- bold  This will be in boldface 
- italic  This is italic 
  or 
- typewriter <tt> This will be typewritten </tt>

LISTS: 3 Types of lists
Ordered, unordered, description

**END OF LECTURE** **********

**JANUARY 19, 2001**

Last Day
- talked about HTML

Today
- more HTML
- special characters, lists, tables
- Design issues for webpages

htaccess remarks:
1. DirectoryIndex myfile.html
since .htaccess file specifies things for current dir
you only have to give filename

2. ErrorDocument 500 http:// complete URL
↑ ↑
no space space
You need to provide the complete URL for document to load up properly.

Note: http://www.math.ucla.edu/~cpollett
the way srm.conf works is that it automatically goes to the public_html.

**More HTML**
Special Characters:

| | | |
|---|---|---|
| < | prints a less than character | (<) |
| > | prints a greater than character | (>) |
| & | prints an ampersand | (&) |
|   | nonbreaking space | |

ex: hi  there

Lists:
3 Flavors:
1. Unordered List
2. Ordered List
3. Definition List

1. Unordered List
Looks like:
- My first important point
- Another pt.
- still another point

Would be written in HTML as:
<ul> ← (start unordered list)
(list item) → <li> My first important pt.
</li>
<li> Another point
</li>
<li> still another pt. </li>
</ul> ← closes unordered list

A. Can change char printed before each item.
<ul type = "circle">
↑
(rather than • would be o)
disk-default
B. can change at any item too.
<li type = "circle"> hi there </li>
this prints:
○ hi there
(all subsequent items will also have a circle unless you say otherwise)

2. Ordered List:
Looks like:
1. Great
2. Okay
3. Marginal

HTML Code:
<ol>
<li> Great </li>
<li> Okay </li>
<li> Marginal </li>
</ol>

A. Just like unordered list there are different types of ordered lists.
A.
<ol type = "A"> "A" would list things B.
C.
can also place "I" → this prints I.
II.
III.

type = "1"
prints → 1.
2.
3.

type = "a"
prints → a.
b.
c.

type = "i"
prints → i.
ii.
iii.

B. Can nest lists.

```
<ol><li> 1st pt.
        <01 type = "A">
            <li> 1st subpt.
            </li>
        </01>
    </li>
</ol>
```

this prints:
1. 1st pt.
   A. 1st subpt.

3. Definition Lists

Looks like:
ham: 1A type of meal made from pigs

HTML code:
```
<dl>      ← <dl> starts definition list
<dt> ham: </dt>  ← title of defn.
<dd>  1.a. type of meal made from pigs.
</dd>
</dl>
```

TABLES:

Ex 1:

| My Data | |
|---|---|
| Trial | Distance |
| 1 | 75 |
| 2 | 77 |
| 3 | 73 |
| 4 | |

HTML code:
```
<table width = "100%"
```
fills up the complete width of draw space
```
 border = "10"
```
there should be a 10 pixel "frame around table"
```
 cellspacing = "4">
```
determines how many pixels around each item

Note: HTML code can be on different lines and still be legal!

```
<thead>  <!- - heading for table - - >
My data
</thead>
<tr> <!- - starts row - ->
<td> Trial </td>
<td> Distance </td>
</tr>
<tr><td> 1 </td><td> 75 </td>
</tr> . . .
</table>
```

A. Can make a table with 1 row
```
<table>                in line style
    <tr><td style = "background:  blue">
        stuff
        </td>          change to a different color
        <td style =
    </tr></table>
```

Gives the following effect often seen

| Some lines | Long fixed doc. |
|---|---|

**END OF LECTURE AND SET #2 ***************************

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #3**

**JANUARY 22, 2001**

HW 2 is posted on the web

Last Day: Talked about list & tables in HTML

Today: We will talk about I. anchoring within documents; II. frames
Then we'll begin talking about PERL

**HTML:**

I. Anchoring

A. Naming part of a document so that you can link to it using an anchor

(EX) Suppose you have War & Peace in a single html file. This is a long document so say you want to look at chapter 3 without having to scroll through the whole document. A way to do this is to add a link.

War & Peace

<h3>
Chapter 4
</h3>

To add something to name this part of document use an anchor.

The code is then –
<h3><a name = "Ch1">Chapter 1
</a></h3> ↓
names this part of doc Ch1

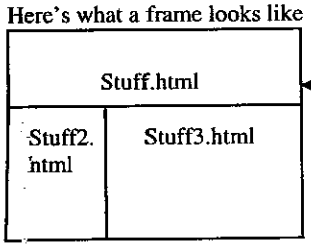
If you want to have a link to this part of document so if you click it you go to Ch1:

<a href = "#Ch1">Link to Chapter 1</a>

You could place this at the top of document so that from the top you could jump to any chapter

II. Frames
Above works well with frames

Here's what a frame looks like




splits browser into regions, each region has its own html document

a. Here's how to make a document with frames

Note: there is no body

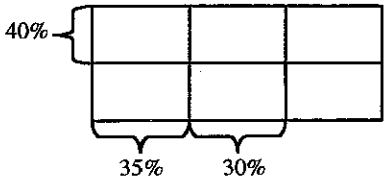
```
<html>
<head> stuff for head </head>
<frameset rows = "40%,*" cols = "35%,30%,*">
```

col1 = 35%
col2 = 30%
rest = *

2 rows
1st takes 40% of height of screen
2nd row remanining amount

3 columns
1st takes 35% of width

This is what it looks like based on the coding above:



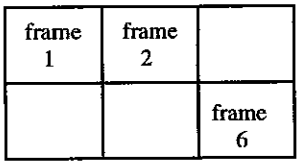

b. How we say which documents are in each region:

```
<frame  src = "frame1.html" name = "f1"  scrolling  = "no" />
```

these are optional

this is to have no scrollbars

```
<frame  src = "frame2.html" . . . . . />
```
(write the rest of the frame lines like the one above)
```
<frame  src = "frame6.html ". . . . . />
</frameset>
```



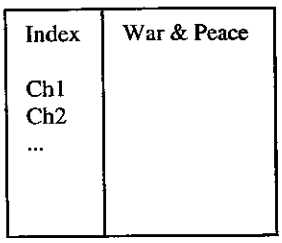

c. How to make a link so that when clicked on, it appears in a given frame

```
<a  href = "my.html"  target = "f1">  my line </a>
```

when clicked on, it opens my.html in f1 frame

Example 2:

Index is the name of this frame




book name of this frame initially loads War and Peace.html

Chapter 4 link might look like:

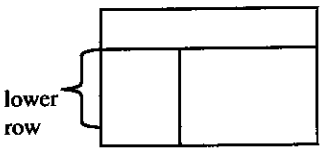
<a href="http://URL/warandpeace.html#Ch4" target="book">Ch4</a>

d. Built-in Targets

___blank → opens new window

___self → open in same frame as the link

___parent → if framesets nested




could use a first frameset to split into 2 rows – then split 2nd region into 2 columns

The code would look like:

```
<frameset rows = "40%,*">
    <frame . . . .>
    <frameset cols = "20%,*">
    <frame . . . .>
    <frame . . . .>
</frameset>
</frameset>
```

if the link was in this frame then __parent would replace the lower row

___top → replaces whole browser window with document contents

⇒ That's about it for HTML for now. Let's talk about PERL

**PERL**

Is a language which is useful for system administration and server side. dynamic content for web pages

very useful for string manipulation and also for manipulating files and directories

It's a scripting language.
Usual scripts are called "shell scripts" and are basically a bunch of UNIX/DOS commands put in a file.

Example: Shell Script

```
#! / bin / csh    ← says which shell
cd    / usr / local

ls    ← this script would print out contents of / usr / local directory
```

To run a shell script

Type at prompt:
> source scriptname

Similarly a perl program looks like

```
#! / usr / local / bin / perl
# now we have perl instructions
    print ("hi there \n");
    print  "hi again \n" ;
```

The # is used for comments in Perl and most script languages

The above is our complete program
Would output

```
hi there
hi again
```

- Notice no main
- Perl functions can be called without enclosing arguments in ( )
- Each line of Perl, like C, end with a ";"

**END OF LECTURE ***************************************

**JANUARY 24, 2001**

HW1 – solution is up
HW2 – is still up

Last Day: we did frames and started talking about Perl
Today: We'll do a more complicated Perl example and look at the different kinds of variables in Perl

A more complicated Perl example:
name.plx

To run a Perl script you could type:

```
perl    name.plx
        ↑ filename
```

Perl Example:

```
print "What is your full name:";
$name = <STDIN>;
```

this line means - read into variable $name from the standard input up to \n

Note: variables begin with $ then like C variable names

```
chomp($name);    ← this line of code deletes any \n characters

if ($name eq "Chris Pollett")    ← tests for equality of strings
  {
    print "hi there, Chris \n";
  }
    elsif ($name = ~/^John/)
  {
    print "Well, hello $name";
  }
    else
  {
    print "Gosh, nice to meet you! \n";
  }
```

notice groovy spelling

matches string beginning with John – it's case sensitive

will evaluate to whatever $name is

Declaring an array:

```
@names = split (/\s/,$name);
```

this is an array in Perl

a pattern that matches any space characters. (ie. space, \t, \n)

splits name into parts at space characters and stores the result in @names.

To print an array & its elements

```
print "From what you said \n";
print "I'm guessing your first name is"
print "$names [0]. \n And your";
pring "last name is $name [1]. \n";
```

this is to access one element in the array

This prints out:

```
What is your full name?
John Smith
Well hello, John Smith
From what you said
I'm guessing your first name is John
And your last name is Smith
```

End of Perl Example.

Variables in Perl

Begin with a special character and then followed by a C-like variable name

Example:

```
$a      ← scalar variable
@a      ← an array
%a      ← hash
\ $a    ← a reference to a variable
```

Let's look at the first three types in more detail

1. Scalars
   Examples: $hi, $bob.
   They can be either numbers or strings
   In reality, though everything is stored as a string

A. Numbers

$a = 5; ← example of the way a number would be stored as string "5". However unlike arbitrary strings, if strings digits are all numbers. You can do +, *, etc. . .

$b = 1.25; ← you can also store decimals and negative numbers

$c = -7E25; ← which means $-7x10^{25}$

$d = 037; ← means 37 in octal 3x8x7 = 31

$e = 0xff; ← hexadecimal is 255

Operations on Numbers
Like in C++
+, *, /, **, =, ==, !=, <, >, <+, >=, +=, -<, ++, - -, . . .etc
↑
to raise to a power
(ex: 2**3 = 8)

B. Strings Sequences of characters

EX 1:
$class = 'It\'s great!';
single quotes mean don't try to evaluate string for variable
$class above gets string → It's great!

↓ single quotes

$class = 'It\'s great! $a';
this prints It's great! $a

↓ double quotes evaluates string

$test = "You know $class";
prints – You know It's great! $a

Special characters for strings are just like C++
Examples:
\n → new line
\t → tab
\\ → \
\" → "
\' → '

Operations on strings:
eq → checks for equality
ne → checks for inequality of strings
=~ → matches patterns

$a = "hello". "world";
↑ period performs concatonation
$ a gets value helloworld

Arrays a little like lists in LISP from PIC 15

Example:
@myarr = (1, 2, 4);

creates an array with 3 elements
$myarr [0] = 1;
$myarr [1] = 2;
$myarr [2] = 4;

$# myarr is 2 – the index of last elements in the array

**END OF LECTURE** **********

**JANUARY 26, 2001**

Announcements:
Practice Midterm is posted on the web.
Homework 1 will be b ack later today.

Last Day- We were talking about variables in PERL

Today- We will be talking about arays hashes & control

**Structures in PERL**

You can get keyboard input into an array use the following code:
@f = <STDIN>;

What this does, is it gets lines from standard input until ^D received.
(^z on arn 98; ^C on old NT)
Each line that is inputed goes into a different element.

For examply: If the input is bob
sally
fred
D

then,
@f= ("bob", "sally", "fred");
↑ ↑ ↑
$f[0] $f[1] $f[2]

Manipulating Arrays:

() → empty array

Example 1:
chomp (@f); | this code removes returns from elements of array

Example 2:
$a = 4 | this is the same as (4,3,4)

$c = @b; | #c is index of last element

@b = ($c, @b, 5, 6);
↑
|

replaces this with its elements. (2, 4, 3, 4, 5, 6)

Example 3: Creating arrays of strings

@d = ("there", "hi");
To save typing, use quotes all the time.

You could aso type:
@d = qw(there, hi);
↑

qw means quote word

Perl has built-in methods for removing or inserting first or last elements of an array.

```
$a = pop(@d);
```

removes last element of @d and puts in $a.
Now,

```
@d= ("there")
@a= "hi"

push (@d, "bob");
@d becomes ("there", "bob")
```

shift/unshift – are like pop & push but work with first element of array.

Built-in Arrays

@_ ← array of arguments to a function.

@ ARGV - the arguments on the command line used in running perl script.

>myscript 300 400

After typin gthis on command line @ARGV is (300,400)

Some Built-in functions on arrays

```
@c = ("you", "there", "sit");
reverse (@c);
```

Now @c is ("sit", "there", "you")

```
sort @c;
```

this code sorts it in alphabetical order. In this example they are already sorted so @c is unchanged.

Hashes

Perl also has an efficient facility for storing tables called hashes.

| Employee | Salary |
|---|---|
| Frankie | 1000 |
| Sally | 1500 |
| Johnny | 500 |

We could use a two dimensional array to store this data but then it would be hard to say: "Find how much Jenny was making without knowing which row she was in".

Could use a hash variable instead

% emp

```
$emp {"bob"} = 5000;
$emp {"alice"} = 6000;
$emp {"john"} = 2000;
```

Looks like:

bob 5000
alice 6000
john 2000

```
print $emp {"alice"}. "\n";
```

it prints out:

6000

Suppose you had an array

```
@a = ("a", 1, "b", 2);
%myh = @a;
```

Now, if you were to write the following code:

```
print $ myh {"b"};
```

it would print
2

```
@arr2= %myh;
```

@arr2 would look exactly like @a

To get an array of keys

```
@mykeys = keys (%myh)
```

@mykeys would equal ("a", "b")
↑
can't have duplicates

```
@myvals= values(%myh)
```

@myvals would then be (1,2)

Control Structures in Perl

if/elsif/else ← we've discussed this already

while ( ) { } - works like in C++

do { } while ( ); - like in C++

foreach - used to go through each element in the array

Example:

```
foreach  $key  keys(%myh)
{
     print $key. "\n";
}
```

key → ("a", "b")
goes through this array one by one and does what's in { }

**END OF LECTURE AND SET #3**
**********

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING**
**PROFESSOR POLLETT**
**SET #4**

**JANUARY 29, 2001**

Midterm is on Friday
We will go over practice midterm on Wednesday
HW3 will be up later today

Last day – We were talking about control structures in Perl

Today - Finish control structures
Talk little about how to read from a file
regular expression functions in PERL

dos2unix myfile myfile
unix2dos myfile myfile

set carriage return for the correct system ^M

Control Structure:

```
$a=0;                    This outputs:
until ($a>5)                    0
{                               1
    print "$a\n";               2
    $a++;                       3
}                               4
                                5
```

The above coding loops until parenthesized expression holds.

Now we will talk a little about reading from files

Reading from files:
<> - diamond operator

Example: Suppose you have a file myfile.plx

.plx – common extensions for single files with perl stuff
.pl – perl library
.pm - perl module

```
#!/usr/local/bin/perl
while(<>)
  {
    print $_;
  }
```

If you type –

perl myfile.plx file1 file2

then <> will read file1 line by line and print result to screen. ($_ has contents of current line). When file is done, do the same for file2. When file2 printed, in this case there are no more files and the program stops.

Let's suppose you don't care to get the file names from command line
You can type:

```
#!/usr/local/bin/perl
@ARGV = ("file");
    while (<>)
  {
    print $_;
  }
```

Now above program would print out file

Regular Expressions:

Regular expressions are collections of strings which are very simple to recognize. Very simple means in one scan of a string, using only constant memory, can tell if belongs to given collection or not. The collection of strings of a given regular expression is called a regular language.

Why regular expressions are interesting:
1) Lexical analysis phase of program compilation
Lexical analysis is where program broken into various names and keywords

2) Searching through documents, lists of tables, lists of processes

In Unix there is a command to search for a given regular expression called grep

Example:

```
ps uax | grep mysql
```

↑ pipe takes output of ps and sends result to grep.
mysql is the regular expression
Above code prints out only lines which contain mysql

Perl has built-in methods for working with regular expressions. Formally a regular expression has a fixed alphabet, say ASCII, and any symbol from alphabet is a regular expression. In addition,

If $e_1$ & $e_2$ are regular expressions,
so are $e_1 \cup e_2$ ← union
$e_1e_2$ ← concatenation begins with something in $e_1$ followed by something in $e_2$
$(e_1)^*$ ← zero or more copies of $e_1$

Perl can express all of the above types of regular expressions. Perl's syntax slightly differs, however, from the formal definition.

Perl's regular expressions

Example:

```
$a =<STDIN>;
```

```
if(/abc/)
    print "That contained abc";
```

abc is the perl regular expression.
The perl regular expression is between slashes.

Other special match characters

. ← matches any single character other than newline
Example:
/s./
This code matches any string containing an s not followed by a newline

[ ] ← can select only one of the characters in the regular expression
Example:
/a[abcde]/
The code matches any string containing an a followed by only one of a,b,c,d,e

^ ← matches regular expressions at the beginning of a string
Example:
/^[abc]/
matches strings that begin with a,b, or c
The above code would accept the following as matches:
alphabet, bob, cat
It would not accept dog as a match

/^[a-z]/ ← must begin with a lowercase letter

/[0-9\-] ← matches strings with a 0,1,2, ..., 9, or a -- sign

\d ← matches a digit

\w ← matches alphabetic letter

* ← zero or more occurrences of expression

+ ← at least one occurrence

exp$ ← string must end with expression

**END OF LECTURE** **********

**JANUARY 31, 2001**

Homework 2 solutions are up.
Today – we will talk about the practice midterm and actual midterm

Actual Midterm
There will be 5 problems
1 problem will come from the practice midterm.

To Review
1. Make sure you can do all the problems on the practice midterm without hestitation
2. Review HW1 and HW2
3. Review Notes
4. Look at book

No baseball caps or hats
No cell phones
Must bring student ID
There is a 2pt. penalty for beginning too early and ending too late.

Practice Midterm
1. Write an .htaccess file that password protects bob.html using the password file /usr/local/bob/htpasswd.

```
<FILESbob.html>
AuthUserFile /usr/local/bob/htpasswd
AuthType Basic
AuthName "Enter your password"
require valid-user
</FILES>
```

2. Give Perl Regular Expression
a. String begins with a or b ends with a C.
/^(a|b)(.|\s)*C$/
the period means any string of characters
the line between a and b means "or"

When ^ is at the beginning of the expression is when it should mean the start of a string match.

When the ^ is like /a^/ should be interpreted as any string that contains a or the caret symbol.

b. String containing the two words Chris and raise.
(Note: This can mean that raise can come before Chris in a string and vice-versa)
/(Chris(.|\s)*raise)|(raise(.|\s)*Chris)/

c. Strings having three or more words
(Note: by 'words' professor means strings from a-z A-Z)

/\s[a-zA-Z]+\s(.|\s)*\s[a-zA-Z]*\s[a-zA-Z]+(.|\s)*\s[a-zA-Z]+/

To search for phone numbers you can write
/\(\d\d\d\) \d\d\d - \d\d\d\d/

Write a style sheet which makes the body background white, paragraph should appear in b lack, and h1 headings in italics.

```
body{background : white}
p{color : black}
h1{font-style : italic}
```

```
chomp($line = <STDIN>); #Now $line has a line from STDIN with
                        no trailing \n
    @word=split(/\s/$line);
    @rev_words=reverse @words;

    $flag=true;
    foreach $word(@words)
    {
        if($word ne (shift @rev_words))
            {
                $flag=false;
            }

if($flag)
{
    print "Yeah!";
}
```

**END OF LECTURE AND SET #4** **********

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #5**

**FEBRUARY 5, 2001**

Instructions on how to submit HW 3 has been added to web page
Midterm
avg was 76% out of 25 pts.
median 78%
If you scored less than 50%, I would worry.
Last time that I said something new, we were talking about regular expressions.

Today – Show the first CGI program
Talk about functions in Perl.

Example of CGI Program

**myfirst.cgi**

The extension, .cgi, is used to tell server to execute the contents of this file and send results back to the requester. The requester is typically a browser.

**CGI** – Common Gateway Interface

The interface is simply the running of certain "specially marked" programs and the sending of the results back to the document requester. (sometimes these files need to be in a special place like a cgi_bin directory.)

**#!/usr/local/bin/perl**
If you don't have this line it won't execute you script. This coding tells server to fork a process and execute script in process and sends output to requester (Don't need it if you have mod_perl which is an apache interpreter for perl.)

**start_html;**
This code calls this function

```
head "myfirst.cgi";
begin_body;
print "<b> Hi there </b>\n";
end_body;
end_html;

# here's where we write the functions
sub start_html
{
print "Content-Type: text/html\n\n;

# tells browser this is an html document
print "<html>\n";
}
sub head
# notice no list of arguments
{
    print <<END_HEAD1;
        <head>      everything between these two tags will
        <title>     be printed as formatted
    END_Head1
    print "$_[0]";
    # the 0th item passed to head function, these items stored at _
    print <<END_Head 2;
        </title>
        </head>
    END_Head 2
} #End of head function

sub begin_body {print "<body>\n";}
sub end_body {print "</body>\n";}
sub end_html {print "</html>\n";}
        #End of myfirst.cgi
```

So if we execute this program.
The following will be printed out:

```
Content-type: text/html

<html>
<head>
<title> myfirst.cgi</title>
</head>
<body>
<b>hithere</b>
</body>
</html>
```

Before doing more with CGI in Perl will talk a little more about functions in Perl and files

More on Functions
Can view function arguments in Perl as lists and arrays.

Example

```
@a = ("man", "has")
squish unisex "no" @a "gone" "before";

# this function gets an @_ that looks like: ("no", "man", "has",
"gone", "before")

sub squish
{
    $squished = join("+", @_);

# The "+" within the parentheses joins together elements of
# @_ in a single string, where elements are separated by +
```

```
$ squish = ~s/man/one/g;
# the code replaces each occurrence of man with one

print $squished "\n";
```
} End of code for example1
Above would output:

**no + one + has + gone + before**

Hashes
How hashes are passed

```
$myhash {bob} = 3;
$myhash {sally} = 2;
myfun %myhash;

# @_ for myfun looks like: ("bob", 3, "sally", 2)
```

Return values from functions

You can return scalars or arrays
(hashes get converted to arrays)

Example:
```
@a = send12arg0 "bob";
sub send12arg0
{
return (1,2, $_[0]);
}

# this would assign

@a = (1,2,"bob");
```

**END OF LECTURE** **********

**FEBRUARY 7, 2001**

Will talk about Bonus in office today at 1:30 & 3:00

Last Day – functions in Perl passing & returning values from functions CGI programming.

Today – files in Perl and local variables

Files in Perl

Example: Program to write out an array to a file of your choice.

```
print "Please enter array of values followed by ^D";
@arr = <STDIN>;
write_array(@arr);
            ↑
        # @_ will look like @arr
sub write_array
{   my $name;
        ↑
    # declares $name as a local variable
    # at this point $name is undefined
    my $app = false;
        ↑
    # can do assignment w/ local vars
    my ck;
    print "Enter a file name to write to:\";
    chomp ($name = <STDIN>);
    print "Would you like to append or overwrite (a/o)?";
    $ck = <STDIN>;
    if ($ck = ~/^a/)
        {$app = true;}
        # checks if the first character in $ ck is an a

    $name = "> $name";
        # when you open file handle,
        # a single > means write to file handle

#stop coding for now to talk about
```

append and output
Note: Aside redirects in Unix
To send output to myfile use the following code

ls > **myfile**

To append output to myfile
ls >> **myfile**

#continue with coding

```
if ($app == true)
{$name = ">>$name";}

open (OUT, $name) || die "There was an error $!\n";
                      ↑
```

> myfile would mean open myfile for writing >> myfile

die causes program to exit & writes "There was an error $!\n" to STDERR

$! is the current STDERR message

```
for each $elt @_
{    print OUT $elt;
}
close OUT;
}    # end write array
```

**OUT is a filehandle, you can think of it as a pointer to the stream disk we are writing to.**

**Names of filehandles do not begin with $**

Output of the above code
```
Please enter array of values followed by ^D
1
2
3
4
Enter a filename to write myfile
Would you like to append or overwrite (a/o)?
0
# if user chose to overwrite
# $name = ">myfile"

# if user chose "a" to append then
# $name = ">>myfile"

# since user chose to overwrite the file
# the file opens and
# 1
# 2
# 3
# 4
# would be written to it.
```

**Reading a file**

**To read a file could do:**

```
open (IN, "myfile");
while (<IN>)
{ print $_;}
close IN;
# notice no > or >>
# so IN will be a filehandle for reading
# $_ sets current line if <IN> evaluated to true
```

Remark

```
print "hi there\n";
# really an abbreviation for
print STDOUT "hi there";
```

Remark 2

my gives us local var's, we can also have semi local variables

**Example:**

```
my $test=1;
bob();

sub bob
{ print "$test\n";
}
```

The output of the above code would just be a newline since it's local to bob. and $test undefined within sub bob

my $test=1; is a file wide local variable, like a local variable to main() in C++

if we replaced
**$my $test=1;**
with
**local $test=1;**

then the above would print
1 <ret>
and $test would be available to any function called from the enclosing piece of code.

**Example:**

```
local $test;
sub bob
{
     local $a=1;
     mary();
}

sub mary
{
     print $a;  # could print one
}
```

**END OF LECTURE ***************************************

**FEBRUARY 9, 2001**

Last day: We were talking about files in Perl.
Today: Talk about directory and file manipulation in Perl.

File Tests
Sometimes we would like to know things like whether a file exists before trying to read it, etc.

To check file existence

```
if (-e "myfile")
     {print ("myfile exists");}
```

Typical application would be if file exists open it and read it otherwise maybe look in some other file.

Other file tests you can do

**-w** Is file writable?
**-r** Is file readable?
**-x** Is file executable?

**-d** Is file a directory?

Example of –d

```
if (-d public_html)
{chdir ("public_html");}
```

Checks if there is a directory in the current directory named public_html. If so, switch to that directory.

**Deleting Files In Perl**

Example:

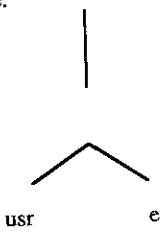
```
print"What file would you like to delete?";
chomp ($name=<STDIN>);
unlink ($name) || warn "Can't delete $!\n";
```

**unlink** deletes a file
**warn** produces a warning on STDERR but doesn't exit program.

To delete all files that end in .txt do
**unlink <*.txt>;**

To rename a file
**rename ("me","you");**
This changes the file named me to be named you.

**Links and Symlinks**
Often the case where you would like to have a given file in lots of people's but only keep one copy of the file. Also, sometimes we would like the directory on your machine to be something other than a tree.




**Hard Link (for files)**

```
link ("/usr/local/people","/home");
```

This creates a link to the file
/usr/local/people named home in the /directory

Link only files
For directories use symlink

```
symlink ("common_file","users_copy");
```

Example:

```
symlink ("/usr/local/java","java");
```

Equivalent to the above in Unix is,
**ln**
**ln –s**

More file manipulation commands

**mkdir ("bob",0777)|| die "cannot make directory");**
Creates a directory named bob which is rwx by everyone.

Note the use of octal

To remove a directory:
**rmdir ("bob");**

To change modifications on a directory
**chmod (0755,"bob");**

Directly using UNIX commands in Perl

Example 1: Can use the system function
**system ("ps uax > processes");**
This would list to the file named processes given by ps.

Example 2: Use back quotes

- ' This is an apostrophe
- ` backquote

**print "the date is ".`date`."\n";**
runs the UNIX command date which prints date.

**END OF LECTURE AND SET #5 ***************************

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #6**

**FEBRUARY 12, 2001**

Corrections
(1) Reading Assignment last week meant Ch. 17 not on Ch. 16
(2) Hard links for files
Soft links for directories.

With a hard link, as long as, one hard link to a file still exists, the file data won't be deleted.
With a soft link you can delete the directory or file associated with links.

**Today**
DBM hashes quick and easy way to store data
server side includes
If we have time we'll start talking about forms.

**DBM hashes**
A convenient way to store hash tables to a disk

Example:
→ **dbmopen (%myhash, "mydb", 0755);**

**%myhash** – will be the name of the table stored in mydb for this session
**"mydb"** – use DBM stored in mydb.page and mydb.dir
**0755** – if it doesn't exist create with permissions

→ **$myhash {"a"} = "b";**
stores a value into DBM

→ **dbmclose(%myhash);**
how data will be stored on disk
Ok, let's read a DBM now.
**dbmopen(%myhash, "mydb", 0755)**

To read in all the values do
**while( ($key, $value) = each(%myhash))** |for each row in %myhash return a $key, $value array.

```
print "$key has value $value\n";
}
dbm close (%myhash);
```

In this case you would print out:

a has value b.

Remark: Say you wanted to skip row with key bob in above example
You could do this by adding line.

```
(if($key eq "bob" {next;}
dbmopen(% myhash, "mydb", 0755)
while (($key, $value) = each(%myhash))
if ($key eq "bob") {next;}
{
    print "$key has value $value\n";
}
```

Deleting Rows in DBM
To delete a row from a DBM when it is open could do;

**delete %myhash ("a");**

in the parenthesis you place whatever key you want to delete.

**Server Side Includes**
A simple way to add dynamic server side content to an HTML page

Example: if you add following line to the body of an HTML document it would print out the last time file1 modified;

**<! - - #flastmod file = "file1" - - >**

If <!- - SSI not enabled it's treated like a comment.
The # says what follows an SSI directive.

General Format of an SSI
**<!- - #directivename tag1 = "value", tag2 = "value". ... - - >**

Other directives
**<!- - #include file = "myfile.html" - - >**

Adds a current location in HTML document the contents of myfile.html. This is useful if you want a common footer to a collection of HTML documents. This way if the footer needs to be changed only have to change in one place.

**<! - - #exec cmd = "ls" - ->**
performs ls on current directory, prints results into current page
cmd indicates to server that what follows the equal sign is some shell command. In this case, we did ls but could do anything.

**<!- - #exec cgi = "my.cgi" - - >**

If you don't put a URL it assumes current directory
The above line of code executes my.cgi and puts results into current document.

**<! - - # echo var = "REMOTE_HOST" - - >**
Prints out value of $ENV {REMOTE_HOST}

**<! - - #if "&&FORMDATA&& = = "bob" goto label1 - - >**
Above says look at value of FORMDATA variable, if it is equal to bob goto label1
**<! - - #label = "label1" - ->**

**Forms**
How to get data from client

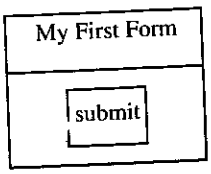
Example: including a simple form into an HTML document

```
<html>
<head><title> MyFirst Form </title></head>
<body>
<form method = "GET" action = "my.cgi">
```

"GET" appends data to URL
action says what URL to send form data to

```
<input type = "submit">
</form>
</body>
</html>
```

Looks like:




When submit button is clicked data sent to my.cgi which is executed.

**END OF LECTURE** **********

**FEBRUARY 14, 2001**

Announcements:
Should relook at HW4
need XBitHack on in .htaccess

SSI works automatically for .shtml files
To get SSI to work with .html extensions need to put **XBitHack on** in .htaccess file

Last Day:
We talked about Server-Side Includes (SSI)

Today:
making forms
processing forms

Example: HTML FORM

```
<html>
<head>
<title> My 2nd Form </title>
</head>
<body>
<form method = "GET" action = "my.cgi"/>
```

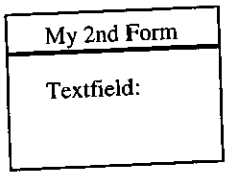
#how will process form, GET or POST. In this case we **GET**
**#action = "my.cgi"** is what handles the form

#The most common thing with forms to make layout look good is to use a table

```
<p>
    Textfield:
    <input type=text name=text1 size=20 maxlength=80>
```

# **size** sets how many characters wide can show up
# **maxlength** sets the max number of characters the textfield can accept
# if you added **value** attribute, could give a default value in the textfield

So far this is what the form looks like

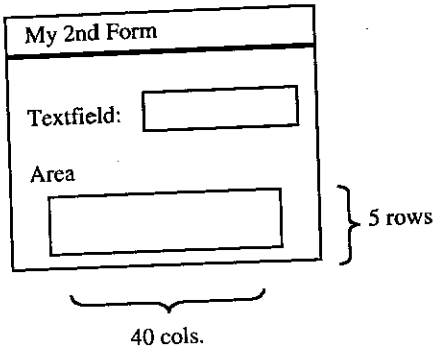



```
</p>

<p> Area:</p>
<textarea name="area" rows="5" cols="40">
</textarea>
```

putting stuff in between the textarea tags gives a default value

Now the form looks like:




To add a submit button

```
<input type="submit" name="submit1" value="submit"/>
```

| | |
|---|---|
| **# type=submit** | makes a submit button |
| **# name=submit1** | is optional |
| **# value=Submit** | this is the value that appears printed on the button, it's optional |

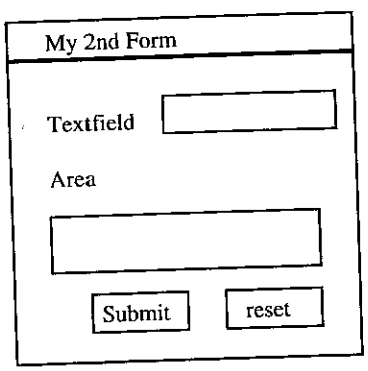
To add a reset button

```
<input type="reset"/>
```

# if you click a reset button it erases all data in form

```
</form>
</body>
</html>
```

Now the form looks like:



What happens when you submit a form?

Get Method

1. First thing that happens, the URL in browser becomes the action of form URL concatenated with a^? followed by $name_1 = value_1$, & $name_2 = value_2$, & . . .

For above,
http://something/my.cgi?text1 **=blah &area= blah2&submit1=Submit**

2. Client sets the server **$ENV{REQUEST_METHOD}** variable to GET

3. Client sets **$ENV{QUERY_STRINGS}** to what is after? above.
For this example, it would set it to
text1=blah&area=blah2&submit1=Submit

So my.cgi script could look at **$ENV{QUERY_STRING}** and figure out what was sent.

Slight Problems:
query string does some encoding to handle special characters like &, =, space, etc. . .

Solution:
use someone else's code to do conversion of Query_String
The most commonly, used is CGI.pm. It's kind of big so we will use books library **subparseform.lib** (this library works for both GET & POST method)

Example my.cgi which just prints out names and values it was passed.

```
#!/usr/local/bin/perl
require "subparseform.lib";

# this loads the library.
# require just loads file but doesn't check if the file compiles until it's used.
# use checks if the file compiles before it loads it.

&Parse_Form;
# runs program in library
# produces a hash % form data with name value pairs

print "Content_type: text/html\n\n";
print "<html>
        <head><title>\n";
print "my.cgi </title> \n";
print "</head><body>\n";

while (($name, $value) = each(%formdata))
{
     print "<p> $name has value $value \n<p>\n";
}
print "</body></html>\n";
```

| my.cgi |
|---|
| text1 has value blah |
| area has value blah2 |
| submit1 has value Submit |

The Limits to GET Method
1. only send 1 kb of data
2. user sees things being sent
POST can handle unlimited data

Advantages of GET Method
The URL can be used to retrieve processed data without the need of filing in form

**END OF LECTURE** **********

**FEBRUARY 16, 2001**

Last Day
We began talking about forms.
Talked about the GET method of sending form data

Today
Talked about POST method
Then talk about some common elements

The Drawback of the GET method was that you could only have 1 kb of form data and all of this data is visible to user.

POST method allows one to send arbitrary amounts of form data.

**How does this work?**
1. When form submitted, client sends info so $ENV{REQUEST_METHOD} is set to POST (HTTP Post Command used).

2. Client also sends how many bytes of data client has that the server might want to look at. This is stored in $ENV{CONTENT_LENGTH}

For a PERL script which is run by server. STDOUT writes over to client. Similarly, STDIN reads from client.

3. To get the form data from the client (using PERL) server just reads.

**read (STDIN, $buffer, $ENV {CONTENT_LENGTH})**

**$buffer** will get the posted data
**CONTENT_LENGTH** is the number of bytes to read.

Once we've got this posted data we can then parse and do whatever we want with it.

The format of how name value pairs stored is the same as GET

**Parse _ Form Function**
In subparseform.lib can be called in exactly the same way to handle posted data. **%formdata** hash will be the result of this function

**Other Common Form Elements**

Radio Buttons

Example:

Male
Female

Radio Buttons are used in situations where we want the user to select only one out of a group of options.

Here's the coding for the above example –

```
<input type = "radio"  name = "sex"  value = "M">Male<br>
<input type = "radio"  name = "sex"  value = "F">Female<br>
```

Since they have the same name, their in the same radio group.
Only one of these two values can be selected.

When the form is transmitted only the selected value will be sent with name sex.

Checkboxes
Allows you to pick multiple entries from a list of items.

Example:

Lunch Menu

- ☐ Fresh Salad
- ☐ CheeseBurger
- ☐ Dessert

The code is as follows:

```
<p> Lunch Menu </p>
<input type = "checkbox"  name = "lunch" value = "Fresh
Salad"> FreshSalad<br>

<input type = "checkbox"  name = "lunch"  checked =
"checked"  value = "cheeseburger"> cheeseburger<br>

<input type = "checkbox"  name = "lunch"  checked =
"checked"  value = "Dessert"> Dessert <br>
```

Selection Gadgets
Example:

Choose a month:

| |
|---|
| January |
| February |
| March |

Select Gadgets can be used to select just one or many.
(Default is just select one)

```
<select name = "month"  size = 3>
```

size = 3 sets how many options visible at a time.

```
<option> January </option>
<option> February </option>
        .
        .
        .
<option> December </option>
</select>
```

value of month will be which one selected.

END OF LECTURE AND SET #6 **********

LectureNotes

Winter 2001

Copyright 2001

**PIC 40**
**PROFESSOR POLLETT**
**SET # 7**

**FEBURARY 21, 2001**

• Announcements
Friday and Monday Professor will be substituting for PIC 20B during his normal office hours. So he will be holding office hours on Friday at 1:00pm

HW 4 will be collected at 4pm

Last Day
continued saga of forms

Today
Talk about forms and also talk about databases & SQL

Form Encoding types

Forms we've seen so far use MIME – encoding application/x-www-form/urlencoded

Forms which send files should use multipart/formdata encoding since more efficient for sending large amounts of information.

Example

**< form method = "POST" action = "my.cgi" enctype = "multipart/formdata">**

sends data like

_ _ _ _ _ _ ( some random numbers) ( a newline)
(30 dashes)
Content-disposition : formdata; name = " (place the name of the form variable between the quotes) "

file data

_ _ _ _ (followed by random character string)

This get form to take a file from client use:

< **input type = "file" size = 20 name = "varname" >**

size 20 is the length available to type filename

Mailed Form

**< form method = " POST " action = " mail to : URL " enctype = " plain/text" >**
**< /form>**

This mails results of form to URL

One last comment on forms
Can have multiple forms in a single html documents.

**< forms method = " POST" action = "my.cgi" >**
**(stuff for first form)**
**< /form>**

if you click on the submit button from this form then only its name value pairs are transmitted and the action is my2.cgi

**< hr >** stands for horizontal rule
it draws a horizontal line
often used to separate these forms

Databases
For large amounts of customer info not practical to use dbm hashes or flat files.

Problems with Flat files/DMB Hashes
1) Slow b/c limited use of index
2) Need to write specialized code to handle integrity constraints.
3) No concept of concurrency control ( i.e., can't manage multiple people wanting same data)
4) Recovery of data if system crashes.

Better to use a well tested commercialized or public domain database to manage large amounts of data since they support above kinds of things.

How To connect databases to the web

The Form and CGI Script are often combined if use server side scripting language. ( i.e. ASP, PHP, JSP)

What a database looks like
Have a collection of table.
Each table has a collection of rows (tuples)
Each column of a given row in a table usually has a name (called attributes)

Examples Library Database

Book

| Call number | Title | Author | Availability |
|---|---|---|---|
| 077 | War & Peace | Tolstoy | N |
| AS9 | Learning Perl | Schuoaztz | U |

You usually underline one of these columns. The underlined one in this example is call number and this is called a key
If you know the key then you have the fixed row

Member

| Member Name | Member ID | Member Phone | Member Address |
|---|---|---|---|
| John Smith | 0001 | (310) 828-5511 | 12 Somewhere |

CheckOut List

| Call Number | Member ID | Data CkOut | Return Date |
|---|---|---|---|
| 077 | 0001 | March 1 | May 1 |

There are three attributes for key.
The arrow from CheckOut List indicates foreign key constraints.

**END OF LECTURE ***************************************

**FEBRUARY 23, 2001**

Last Day
Talked about databases

Today
SQL - Structure Query Language
- language for manipulation databases

The Database that we looked at last day: Library DB

Book

| call Num | Title | Author | Availability |
|---|---|---|---|

Member

| Member Name | Member ID | Member Phone | Member Address |
|---|---|---|---|

CheckOut List

| call No | Member ID | Checked Out Date | Return Date |
|---|---|---|---|

Above without rows of data for each table called a DB schema
Database Schema in SQL
To create a schema in SQL do:

**create database Library;**

standard SQL is not case-sensitive (i.e. Create, create, or Database, database)
MySQL is case sensitive when it comes to names of databases, tables, columns.
MySQL likes statements ending with a semi-colon

To create a table
Example:

```
create table Book
  ( call No      char (3)       NOT NULL,
    Title        varchar (20),
    Author       varchar (15),
    Availability char (1),
    primary      key(callNo));
```

Note:
char (3) must have 3 characters
varchar (20) up to 20 characters
NOT NULL could be no characters

Can also have types
INTEGER
FLOAT
DATE
MYSQL
supports enumerated types.

To delete database or table

```
drop database Library;
drop table Book;
```

Commands to add rows to tables & delete

To add a row into a database

```
insert into Book
values ('077', 'War & Peace', 'Tolstoy', 'Y');
```

To delete a row from Book

```
where Author = 'Tolstoy';
```

This will delete all books that were written by Tolstoy.

We will be interested in Querying DB's in this class

The Basic form of an SQL query.

```
select Title, Author from Book, CheckOut List
where (Book.callNo = CheckOutList.callNo)
    AND (CheckOutList.MemberID = '0001');
```

select Title, Author -- a comma separated list of column values to return from Book, CheckOutList -- the tables, the query is on AND (CheckOut List . . . ); -- You can replace AND with NOT

Since both Book & CheckOutList had a column named CallNo, we had to stick the name of the table then period.

This returns Title and Author of all books member 0001 checked out

Example Learning Perl, Schwartz 1984, Orwell

Some useful abbreviations

**select * from Book;**

* means it output all the columns
Notice that you don't need the where clause.
This returns the callNo, Title, Author, availability of every book stored in Book

Say we like an author but we can't remember his name

```
select Title, Author from Book
where Author like 'J?h%
```

? in the above code matches any single character
% matches any string

This would match
John
Jah Won

It returns title and author of all books matching above pattern

How to connect to the MySQL through hermosa

MySQL is a free database system. Have a copy on Hermosa (other products DB2, Oracle, Informix, INGRES, Access, FoxPro . . . )

To connect to database type at Unix prompt

**mysql - u pic40 - p**

it will then prompt you for a password. Type **pic40**
This account only has select privileges on air transportation database.

After login it prints stuff and gives you a prompt.
To see what databases available
**show databases**

To connect to air transportation
**use airtransportation;**

To see what tables this DB has do:
**show tables;**
For fun you can try to make queries on these tables.

**END OF LECTURE AND SET #7 **********************************

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #8**

**FEBRUARY 26, 2001**

Announcements
HW 4 solutions is up
HW 5 up by this afternoon or else we'll be given an extra day

Last Day
SQL

Today
Perl DBI module (Database Interface)
PHP scripting language

Example: Use of Perl DBI

```
#!/usr/bin/perl
BEGIN {
    unshift (@INC,
        "/net/willow/h1/fa/cpollett/perldb");
}
```

Note:
**BEGIN** - executes before trying to load any modules
**@INC** - is an array used to store paths to Perl modules

```
use DBI;
$dbh=open_db ("mysql", "airtransportation");
```

**"mysql"** is the type of database (could also be oracle)

```
$cur = $dbh → prepare ("SELECT" * FROM Airport");
```

**$cur** is the object that can be used to execute a query
**$dbh** is a global variable set by open_db.dbh is a database handle object
**("SELECT * from Airport")** query lists all row of Airport table

```
$cur → execute( )
```

→ calls the execute method of $cur object

**execute** causes query to be run

```
if ($DBI::err){
    print "Error:  $DBI::errstr\n";
    exit (1);
}
```

The above code checks if there were any errors executing this query

```
while (@row = $cur → fetchrow)
{
    foreach $field (@row)
    {
        print "$field \t";
    }
    print "\n";
} # end of while
```

```
close_db ($cur, $dbh)
```

**while (@row = $cur → fetchrow( ))** goes through the rows returned by query one by one

```
sub open_db
{
    my$dbd = $_[0];
    my$dbname = $_[1];
    my$dbh;
```

specified the database
opens connection to our database

```
    $dbh = DBI → connect ("dbi:  $dbd; dname = $dbname;
host=hermosa", "pic40", "pic40");
```

user password

dbd means database driver

```
if ($dbh){
    print "Error: $DBI::errstr\n";
    exit (1);
    }
return $dbh;
}

sub close_db
{
    $_[0] → finished( ); # done with the cursor named $cur
    $_[1] → commit( ); # not necessary since only read
    $_[1] → disconnect( ); # closes connection
}
```

Some useful variables available in DBI
After we did **$cur → execute( );**
and looked at the rows in
**DBI::rows**
it would contain a number of rows returned by this query

```
$cur → {SCALE}      an array of sizes of fields
$cur → {TYPE}       array of types of the fields in a row
$cur → {NAMES}      array of names of fields
```

Example: Using other commands with DBI

names of fields in MyTable

```
$cur = $dbh → prepare ("insert into MyTable (name, value)
values (?, ?)");
```

everytime $cur → execute ( ) is
run will fill in these two parameters

Say we have
**$cur → execute (bob, 11);**

this adds a row to MyTable  bob would be the name, 11 the value

**$cur → execute (Alice, 12);**
this now inserts row Alice 12

PHP
Another language to learn.
Used for server side scripting and can be directly inserted into an HTML document.

Example: Myfirst.php
if you want PHP to work you need the .php extension

```
<html>
<head><title> My first php program </title></head>
<body><p>
beg of php → <? echo "hello world"; ?>
</p></body></html>
```

print would also work

end of php

In your browser the above code looks like

| Myfirst php program |
| --- |
| hello world |

PHP is like a mixture of PERL & C

Variables in PHP
Variables are of form **$var_name** just like in Perl.

Variables in PHP
Variables are of form $var_name just like in Perl.
PHP is like a mixture of PERL & C
Variables in PHP
However, **$a** could be a scalar, array, or hash
You can cast variable type

```
$a = (array)$b;
```

int or float

Functions in PHP
The functions are slightly different from PERL

Example:

```
function test($a, $b)
```

we need to say the parameters
also the keyword is **function** instead of sub

```
{
    echo "$a $b \n";
}
test (1, 2);
```

would print to the screen

1 2

Difference between Perl & PHP

To write functions in PHP do things like:

```
function test ($a, $b=1)
{
    print "$a $b\n";
}
```

In PHP don't need to specify return values ← **($a, $b = 1)**
Can call the test function by using:  if don't pass $b its value is 1

```
test (1, 2);
```

outputs: **1  2**

In PHP **$a** can be a scalar, an array, or associative array (i.e. . . . hash)

Comparisons in PHP
== behaves like both **eq** and == in Perl

Local Variables
All variables are by default local.
So if you have –

```
$a = 1;
function out()
{
    print "$a\n";
}
```

$a is local to out so undefined

If you wanted to access global $a would do:

$a is local to out so undefined
If you wanted to access global $a would do:

```
$a = 1;
function out()
{
    GLOBAL $a;
    print "$a\n";
}
```

this prints: **1**

Built-in Globals
Any environment variable in Perl is immediately available as a global in PHP. So the value of **$ENV{REMOTE_ADDR}** in PERL is the same as the value of **$REMOTE_ADDR** in PHP

In addition any variable GET or POST by a form is immediately available as a global variable in PHP.

Example

```
    print $elt\n";
}
```

prints out each element of array $array

To create an array
just declare elements
**$arr [0] = s;**
**$arr ["hi"] = "bob";**
now it's like a hash

**$arr=array(1,100);**
creates an array of size 100 with start index 1.

split in PHP
**split ("\t", $string)**

in PERL we did **split(/\t/, $string)**

PHP Database example

To print out rows in Airport table of airtransportation DB

```
<html>
    <head>
        <title> Airports </title>
    </head>
<body bgcolor = "99FFCC">
<h1> Airports in airtransportation
</h1>     start of PHP block
<? $db="airtransportation";
$connect = mysql_connect ("localhost", "pic40", "pic40");
```

above line connects to mysql server

ora_connect machine user password
if oracle etc. . .

```
    $cursor = mysql_db_query ($db, "select * from Airport");
    ?>  stop PHP block

    <table border = "2" width = "100%" >

    <? while ($row=mysql_fetch_row ($cursor))
        {   print "<tr>\n";
            foreach ($row as $field)
            {
                print "<td>$field</td>\n";
            }
        }
```

The above code prints out our table.
**$row** is an array of field values

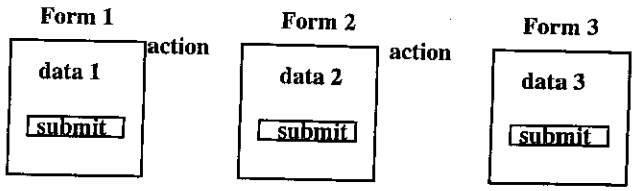
**mysql_free_result ($cursor);**
this gets rid of cursor

**mysql_close ($connect);**
closes connection to the database

```
?>

</table>
</body>
</html>
```

Form Processing
Suppose we have more than one form




variables of form1 were available in creating form 2
However when we create form 3 unless we say to pass form 1 variables explicitly, they won't be passed on.

To pass them on use hidden fields
i.e. in form 2 have a tag

```
< input type = "hidden"
        name = "form 1 varname"
        values = "form 1 varvalue"/>
```

**END OF LECTURE** **********

**MARCH 2, 2001**

Cookies:
Cookies are data the server can tell the client to store on the client's machine

Usually stored in a file cookies.txt

Cookies are sent before the content-type of the document

Ex PERL

```
print "Set_Cookie:  bob=\ "bald\";\n";
print "Content-type:  text/html\n\n";
    .
    .
    .
    Rest of html DOC
```

If server later wants to read available cookies can look at
$ENV {'HTTP_COOKIE'}

this variable looks like:

$name_1 = value1; name_2 = value_2; \ldots$

for each cookie that was set (could have set many cookies)

Could parse using something like

```
%cookies=split (/=^0_U/,$ENV{HTTP_COOKIE'});
```

so cookies {'$name_1$'} = value1;  (approximately)

The client only returns cookies from the same IP address as server requesting cookies (supposedly)

More complicated cookies

```
Set_Cookie:  bob=bald;
    expires:  Thu, 01-AUG-2001
              01:00:00 GMT;
    when to delete cookie

        domain = www.bob.com; ← only return cookie if this is
                                server
        path = /bob ← directory on server where document
```

requested
secure ← use secure connection for cookie

After name value each of above is option (will make harder to parse)
Above restricts the cookie to be returned only if client looking at page
http://www.bob.com/bob

To set a cookie in PHP, before the <html> tag

```
SetCookie ("bob", "bald");
SetCookie ("barb", "redhead", "time",  ,      ,    ,  );
                                  ↑    ↑      ↑    ↑
                                time  domain path  secure
```

To read a cookie in PHP . . .

(like form data all cookie names are global variables in PHP. So for above $bob would have automatically been assigned value bald.

More PERL/PHP distinctions

1) elsif (Perl)  else if (PHP)

switch/case works in PHP

no switch/case in Perl

Could fake in Perl as

```
                          what we switch on array of size 1
                                  ↓
SWITCH:  for each $case ($var)
  ↑      {                    what want to match
unlabeled     if ($case=~/   /)
block         {
                  Do stuff;
                  last SWITCH;
                       ↑
                      exits switch block
              }
              .
              .
              .
              more cases
         }
```

```
SWITCH:  foreach ($var)
    {
        /    /&& do
matches      {    yourcode
$_               last SWITCH;
             }
    }
```

**END OF LECTURE AND SET #8 ****************************

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLET**
**SET #9**

**MARCH 5, 2001**

Last day: pretty much finished talking about PHP & PERL

Remainder of the quarter will talk about Javascript

PHP: portable hypertext homepage preprocessor.

Javascript:
Originally called Livescript and was developed by Netscape. Java came along and seemed sexy so Netscape renamed.

Livescript as JavaScript

Microsoft came along called theirs JScript.

Javascript was submitted to the European Computation Machinery Asoc. for standardization so get ECMAScript.

Javascript is a language that is sent as part of an HTML document to the client (browser). Browser then interprets script to run program.

Remarks
This is different from Perl and PHP which are run on the server.

Why not use same language everywhere?

Good question...
Javascript can be used as a scripting language in active server pages (ASP) document.

Perl – has a client side variety called Perlscript. But need to do something to browser to get interpreter.

Coldfusion – tag system
for some web servers.

Automatically generates javascript for stuff done client-side. (coldfusion is slow)

Javascript is useful to relieve computational burden from server.

Ex.
you can verify properties of forms using javascript.
i.e. digit in credit number are really from possible credit card, etc.

Can make page elts. more dynamic
1) Cycling through a list of images
2) Cause new behaviors for buttons
3) Enhance selection gadgets

**Java vs. Javascript**

Since Javascript programs are usually small text files they are usually a fair bit shorter than the corresponding Java byte code.

So downloads faster → someone waits to see your site.

**Ex Javascript program**

```
<html>
<head><title>
    My first javascript program </title>
</head>
<body>

<hl>
<script language = "javascript"
        type = "text/javascript">
say what follows
is a script
<! - - hide from old browser
            //javascript comment
            document.write ("hi there")
    ; are optional in javascript
    with (document)
    {
    write ("hi again");
    write ("hi 3)
    }
//a with block allows you to take an
//object and calls many of its methods
//without having to go object.method( )

- -> end of javascript

</hl>
</body>
</html>
```

looks like:

My First Javascript
Hi there hi again hi 3

**Alerts & Plugin – Detection**

```
Ex  <html>
    <head> <title> Alert </title>
    <script language = "javascript"
            type = "text/javascript">
<!- -
if(!navigator.plugins["QuickTime Plugin 2.0"])

{alert("Don't have Quicktime!");}

- - >
```

</script>
<no scipt> only print if <script> tag doesn't work
Update your browser you loser
</no script>

If didn't have Quicktime

Don't have Quick Time ← a new window would appear

**END OF LECTURE ****************************************

**MARCH 7, 2001**

Last day: introduced Javascript wrote some short programs
Saw the built-in objects document and navigator
document – has properties of current html page being looked at
navigator – has properties of the current browser

Today: We will go over more of javascript syntax.
Discuss event handling a little more on the javascript object model.

Syntax
To declare variables in Javascript.
var $a, b, c = 2, a;
semicolon is optional.
$a and a are treated as two-different variables.

Like Perl/PHP don't need to declare variables before using them, but it's good style.

Declaring Arrays

```
var $b = [1, 2, 3];
document.write("<p>" + $b[1] + "</p>");
```

need string concatenation in order for it to know that we want the value of $b[1] to be printed.

This code prints 2 to document.

```
var a = Array(10);
```

The above code is an array of size 10 which is empty. We haven't put anything in it yet.

```
var a;
```
Here we have declared a variable without a value.
If you don't give a variable a value it has value NaN

Declaring Objects

```
var O = {x:1. y:2. total:3};
document.write(O.total);
```

This prints a z

Functions in Javascript
Similar to PHP, but don't have to worry about GLOBAL keyword.
Example:

```
$c = 2;
function Hello($a)
{
    document.write($a);
    document.write($c);
}
Hello(z);
```

prints 32

So global variables always accessible. All calls by value.

Comparisons in Javascript like PHP

== !=
works on both strings and numbers.

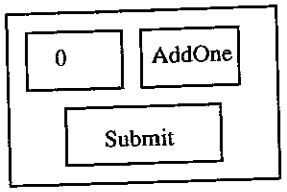
Looping in Javascript
while, for – same as in C, C++, Perl, PHP

```
for (z in 0)
{
document.write(z);
}
```

where 0 object defined above
prints:
1 2 3

```
var a = [1, 2, 4];
for(z in a)
{
document.write(z);
}
```
prints:
1 2 4

Let's look at a more substantial Javascript program




clicking on AddOne button
it increments number

if you try to change the box that keeps count of the number directly it'll pop up an alert message

Another alert box appears
if you try to change the submit button.

```
<html>
<head><title> Javascript Event </title>
<script>
function increment( )
{
document.forms[0].counter.value++;
}
</script>
</head>
```
first form on document
name of element on form
add one to it's value

```
//Notice functions are located in the head of the Doc.
//Alternatively you could have done
document.forms[0].element[0].value++;

<body bgcolor = "white">
<h1>Counter</h1)
<hr>
<form>
<input type = "text"
       name = "counter"
       value = "0"
       onChange = "alert('cannot touch this')"
       readonly = "true">
```

means you can't alter the value because it will return to its orginal value.

```
<input type = "button"
       name = "adder"
       value = "AddOne"
       nClick = "increment( )">
                  ↑
            calls increment( ) if button clicked
</form>

<form method = "GET" action = "index.html"
       onSubmit = "alert('you have clicked me into submission')"

<input type = "submit">
</form>
</body>
</html>
//end of program
```

onChange events
can also be used with selection gadgets.
So if you click on an itme and drop down menu then it automatically process.

If function to handle onSubmit returns true then form is getted or posted.
If it returns false it isn't
Can use to validate forms before sending it to the server.

**END OF LECTURE** **********

**MARCH 9, 2001**

Strings in Javascript

```
var a = "hi there";
  ↑
  now a is a string object
  Has a lot of built-in properties
  Can now access document.write (a.length);
document.write (a.charAt(3));
                ↑                    ↑
                would print t        prints length of string
```

WebTV screws up on charAt
- always returns 0

```
a.substring (1,2);
             ↑
        return 2nd through 3rd char of a
```

- So, a.substring (3, 3) pretty much equiv. to above

```
var myArray=a.split ("t");
document.write (myArray[1]);
     would print "here"
parseInt (string)
     ↓
     converts string into integer
```

More on events in Javascript
Using onChange events with select tags

```
Ex  <html>
    <head><title>select test</title>
    <script>                    name of select tag
    <!- -                       ↓
        function jumpPage (loc)
            newPage = loc.options[loc.selectedIndex].value;
            if (newPage != " "){
                window.location.href = newPage; }
            }          ↓     ↑
```

another built-in Javascript object refers to current window

location current window

specified as URL is at

```
- - >
</script>
</head>          only called if Javascript doesn't work on browser
<body>                ↑
    <form action="gotoloc.cgi"
          method = "GET">
    <select name="location"
          onChang="jumpPage (this.form.location)">
                                reference current form
    <option value = " "
        selected = "true">
        Select topic </option>
    <option value = "my.html">
        MyPage </option>
    </select><noscript>
          <input type="submit"
              value = "Go There">
              </noscript>
    </form></body></html>
```

If have Javascript

pull down to option ← [select a topic] ← if pulled down and released mouse on MyPage would go to my.html

If don't have Javascript

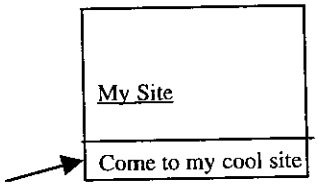
[select a topic]
[Go There]

Fun things to do with links & images onMouseover onMouseout event
Pass mouse over a link and without clicking something

Ex: Something gets printed to status bar

```
Consider link              set what is written on the status line
<a href = "my.html"          ↓
     onMouseover = "window.status
          ↑          = 'come to my cool site';
when pointer for Mouse  return true;" ← need to return true for this
on the link                                to appear
     onMouseout = "window.status=";
          ↑
when move off link      return true;">MySite</a>
```

Looks like . . .




When pointer moves off My Site status goes back to being empty

status bar

MIXING LINKS W/ IMAGES

```
<html>
<head><title> Fun with images </title>
<script>                create a new Image object
    arrowRed=newImage
    arrowBlue=newImage
    arrowRed.src="red.gif"
    arrowBlue.src="blue.gif"
```

LectureNotes

Winter 2001

Copyright 2001

**PROGRAM IN COMPUTING 40**
**PROFESSOR POLLETT**
**SET #10**

**March 12, 2001**

Remark on HW
There was a stray open form tag in Intro ( ).
I have deleted this.

Some people were asking about VerifyDate ( )
To find value of DEPDATE textfield could do:
var $DEPDATE= document.forms[0]. DEPDATE.value

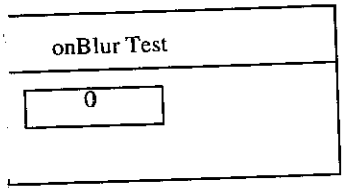
Last Day
Talked about strings in JavaScript and more on eventhandling (mouseOver & mouseOut)

Today
Will talk a little bit more about form verification and then talk about cycling banners and timers

Form Verification
We've seen form verification using onSubmit event.
This is useful if want to verify whole form at one time.
However, suppose we wanted to verify entries one at a time as the user entered them could do this by using onBlur events.

Example 1

```
<html>
<head><title>onBlur test </title>
<script>
    function lessSeven ( )
    {
        if (document,forms[0].counter.value
        {
            alert ('Enter a number less than 7');
            return false;
        }
        return true;
    } //end lessSeven ( )
</script></head>
<body>
    <form>
        <input type= "text"
               name= "counter"
               value= "0"
               size= "8"
               onBlur= "lessSeven ( );">
    </form>
</body>
</html>
```

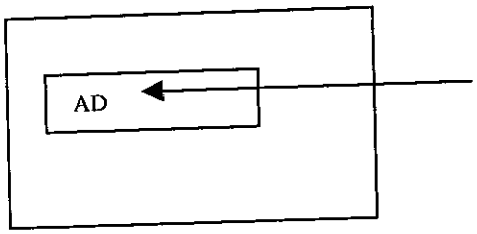



When the user starts typing in here anFocus event is generated.
If then click anywhere else in window an onBlur event is generated.

If someone types in a 7 in the textfield then a box comes up saying "Enter a number less than 7."
But if you entered the letter 'a' the alert message would not pop up.

On 0-6 would not have alert
On a string or letter no alert.

Cycling Banners
You can use a single GIF which has built-in animation or can do it yourself with several GIF and timers




After some time this image changes. We will do it so that the image is a link and this link changes with the image

Example

```
<html>
<head>
    <title> Cycling Images</title>
    <script> <! - -
        ad Images = new A..ray ("banner 1.gif", "banner 2.gif",
        "banner3.gif")

        //some names of GIF filed in current directory
        ad URL = new Array ("yahoo.com", "sun.com",
        "cnn.com");

        //places we go when click on a given banner
        this Ad= 0;
            //current Ad we're looking at

        img Ct = ad Images.length;

        //the above code gets run when document loaded
        //above are global variables.


        function rotate ( )
        {
            if (document.images) //checks if browser supports
          // image objects
            {
```

```
</script> ← so images loaded as soon as page loads
</head>
<body>
<a href = "next.html"
      onMouseover="document.arrow.src = arrowRed.src
                                    ↓
                              name of image tag below
      onMouseout = "document.arrow.src = arrowBlue.src">

make image into an anchor
    ↓
<img src="blue.gif" width="100" height="50" border="0"  name =
"arrow">
                                                          ↑
                                                       name is
                                                referred to fr. above

</a></body></html>
```

**END OF LECTURE AND SET #9 ***************************

```
        if (document.adBanner.complete)
                    ↑
        //have finished loading image
        //name of the image tag
        //we do not  want to cycle through the images before
        //we've loaded them.
        {
            thisAd = (thisAd + 1)%imgCt;
                        ↑
            //goes to next image
            // % allows you to go up to the max amount of
            // images and then starts back to the first one.
                document.adBanner.src =adImage[thisAd];
    }
    setTimeout ("rotate ( )"; 3x1000);
                    ↑
                called in 3000 m sec.
    } //close if (doc.images)
} //end rotate.
```

When we change the image we also need to update what links it points to .
This is the purpose of function newLocation ( )

```
    function newLocation ( )
    {
        document.location.href="http://www"+ adURL [this Ad]";
    }
</script></head>
<body bgcolor= "white" onLoad = "rotate ( )">
    //when body loaded call rotate ( )
<a href = "javascript: newLocation ( ) ">
    //when clicked calls this function
<img src = "banner 1.gif" name = "ad banner">
</a>
    </body>
    </html>
```

**END OF LECTURE **************************************

**MARCH 14, 2001**

Today – will talk about window handling with Javascript
Thursday: TA's will go over 1-5 of practice final
Friday: Will go over 6-10 of practice final

Final is Monday 11:30 – 2:30

How to open a new browser window in Javascript

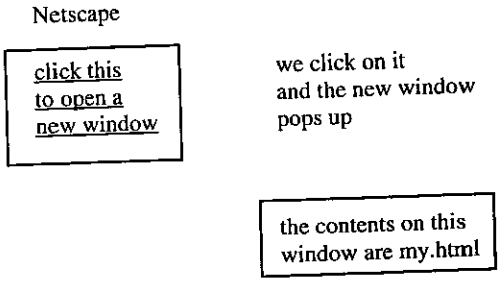
Example:

```
<html>
    <head>
    <title> Opening a Window </title>
        <script>
            function newWindow($URL,$Name)
            {
            $URL is the URL to be opened
            $Name – name of window if want to target in html

myWindow = Window.open($URL,$Name,1width=330,height=250);

            // myWindow – is the new window object
            // Window.open – built-in class
            // 'width=330, height=250' – are all within the same quotes
            }
        </script>
    <body>
        <a href = "javascript:newWindow(my.html,'name')">
        click this to open a new Window </a>
    </body>
</html>
```

Looks like:




Scroll bar
We can add a scroll bar to window and scrolling consider:

```
    function newWindow($URL,$Name,$down)
    {
        myWindow=window.open($URL,$Name,1width=330,
            height=250,left=50,top=60,scrollbars=yes');

    // $down – will control how fast will scroll when timer goes off
    // left = 50 – open window 50 pixels from left of screen
    // top =  60 pixels from top of screen
    // scrollbars = yes – adds vertical & horizontal scrollbars as
    needed

        myWindow.focus( );
        setTimeout("myWindow.scroll(0,"+$down+")",1000);
    }   //end NewWindow
```

myWindow.focus( ) – forces focus into this window

After 1000m sec myWindow.scroll( ) and whatever the value of $down will be called this has the effect of scrolling down the myWindow document $down many pixels.
(In this instance, only scroll once)

myWindow.scroll(0,"+$down+") – the first argument which is zero is the horizontal scroll.
second argument is for the vertical scroll.

Example **Using the name of a window**

Suppose $Name was bob.
If an original document want to have a link which opens in bob could do

```
    <a href = "newpage.html" target = "bob">Open in bob</a>
```

Suppose now in bob window would like to have a link which changes what was in original window

```
    <script>
        function printToOrig( )
        {
Window      opener.document.write("<p>hithere</\p>);
that opened
Bob
        }
    </script>
```

<a href = "javascript:printToOrig( )"> Click here to
print hi there in original window </a>

Original Window

Link Open Bob

Click here
to print
hi there in
original window

Also note in newWindow function
could have had lines like

myWindow.document.write("<p>hi there<\ /p>");

would print to newly opened window
reference to newly oepned window

Closing Windows that are open
Can do things like:

```
if(myWindow&&!myWindow.closed)
{
    check if myWindow      is window
    is not null            already closed

        myWindow.close;
}
```

**END OF LECTURE** **********

**MARCH 16, 2001**

PRACTICE FINAL

1. write .htaccess to get SSI on .html
   hello html print last mod time
   exec helb.cgi

   .htaccess

   HBitHack on
   ---
   hello.html

   ```
   <html><body>

   <! - - #flastmod virtual = "hello.html" - ->
   <! - - include file = "bob.html" - ->
   <! - - exec cmd = "hello.cgi" - ->
   ```

2. write cgi opens file raddresses and write remote address of the person who accessed the cgi.

   ```
   #! / usr / local / bin / Perl

   open (RAD, ">>raddresses");

   print RAD $ENV{'REMOTE_ADDR'};
   for each $k (keys(%ENV)){

   print "$k is $ENV {$k}";
   }
   ```

3. ENCTYPE

<form ENCTYPE = "application/x-www-form-urlencoded">
↑
default

Realname = A+Bug&email = bug@pic
---
<form ENCTYPE = "multipart/form-data">

//transmit larger pices of data

- - - - - - - - - - - - - - - - - - - - 8124

Content-Disposition: form-data, name="Realname"

A Bug

- - - - - - - - - - - - - - - - - - - - 87214

Content-Disposition: form-data; name="email"

bug@pic.
---

```
<FORM ENCTYPE = "text/plain"
        ACTION = "mailto:bug@pic">
<TextArea>
</TextArea>
<input type = "submit">
</FORM>
```

GET puts parameters on the browser's address which are passed to cgi via the "command-line" prompt

POST puts the parameters on the STANDARD INPUT to the cgi

4. Set and retrieve cookie called
   book value = "War & Peace" in Perl

   ```
   #! / usr / local / bin / pen
   print "Set-Cookie: book = War and Peace;"

   //put before MIME header or any HTML

   if ($ENV{'HTTP_COOKIE'}){

       @cookies = split(/:/, $ENV{'HTTP_COOKIE'});

       for each $cookie (@cookies){

       ($n, $v) = split (/=/, $cookie);

       $CRUMBS{$n} = $v;
       }
   }
   ```

   $ENV{'HTTP_COOKIE'}
   → book = War and Peace: user = bug
   crumbs {'book'}

   in PHP

   ```
   setCookie("book", "War and Peace"),

       $book
   ```

5. select title from Book, Checkout
   where
   Book, callno = Checkout.callno
   and Author like "Tolstou"

**LECTURE – MARCH 16, 2001**

Today- will go over Practice FInal Prob 6-10

Final is Monday 11:30 - 2:30 here

Bring student ID
closed Books closed notes.

Format similar to Midterm
One problem on practice final on final
7 problems each worth (5pts)

Practice Final (continued)

6. Write the PHP needed to connect to a MySQL database named airtransportation on the local machine using username pic40 and password pic40
Print out all rows in the table Flight.

```
<html>
<head><title> Flight </title>
</head>
<body>

to open  connection
< ?  $id= mysql_connect ("localhost", "pic40", "pic40");

to prepare query
$result=mysql_db_query ("airtransportation", "select *
fromFlight");

 ?> <table border = "2" width = "100%">
 <?
     while ($row=mysql_fetch_row($result))
{
     print "<tr>"
     foreach ($row as $field)
     { print "<td>$field</td>"}
     print " </tr> \n";
}    // end of while loop
print "</table> </body> </html> \n";
mysql_free_result ($result);
mysql_ close($id);
? >
```

#7 Explain how to use Javascript to check if a text element named fourwide has at least four characters in it before a form can be submitted. Without loss of generality assume four wide is on the 0th form on the document.

In head of document have following function

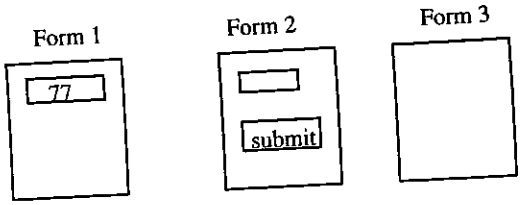
```
function FourCk ( )
{
     var four = document.forms [0]. fourwide.value;
     if (four.length>3) {return true; }
     return false;
}
<form method = "POST" action = "some URL" onSubmit =
"return FourCk ( );" >
          .
          .
          .
<input  type = "submit" >
</form>
```

8. Give an example of the html needed to create a hidden field.

What are such fields good for?

Hidden elts are useful for passing along name value pairs that were collected.

On forms earlier than the current one




name = bob

we would like the
value of bob around when we
generate this page

To make a hidden field do

```
<input type = "hidden"
     name = "fieldname"
     value = "fieldvalue" >
```

for example bob

put this on form 2

bob's value from form 1

**problem with hidden fields is on the exam**

9. Write the HTML and javascript needed to have an image link that alternates b/w the images bob1.gif depending on whether the mouse is over the link or not.

in head of the document

```
var bob1 = new Image;
bob1.src = "bob1.gif";
var bob2 = new Image;
bob2.src = "bob2.gif";
```

in body of the document

name of our img tag

```
<a href = "our URL"
     onMouseover = "document.bob.src = bob2.src;
     return true;"

     onMouseout = "document.bob.src = bob1.src;
     return true;">

     <img src = "bob1.gif" name = "bob">
     </a>
```

10. Write the Perl needed to check if the file hash.dir exists and if so to open the corresponding DBM hash and add one to the value corresponding to the key TOO_SMALL

```
if (-e "hash.dir")
{
     dbmopen (%myhash, "hash", 0755);
     $myhash {'TOO_SMALL'}++;
     dbmclose (%myhash);
}
```

**END OF LECTURE AND SET #10 ***************************