Internet Technologies Notes

Contributor: Shakshi

[Mata Sundri (DU)]

Computer Science Notes

9/July/2019

→ Factorial

```
<html>
<head>
<script>
        function fact()
        {       var i, no, facto;
                facto=1;
                no = Number(document.getEler
                    ById("num").value);
                for(i=1; i<=no; i++)
                {   facto = facto * i;
                }
                document.getElementById("answ
                    value = facto;
        }
</script>
</head>
<body>
        Enter Num: <input id="num">
        <button onclick="fact()">factoria
        </button>
        <input id="answer">
</body>
</html>
```

Array & Arraylist

Page No.
Date

11/7/19

* Java (Array, Arraylist)
* JavaScript
* JDBC
* ISP
* Java Beans

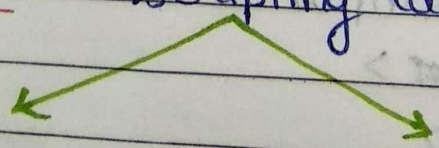
* Jdk - latest - 12.0.1

Lab 1. WAP to declare and initialise an array called "data" of 10 integers.
Do the following -
- find and print the smallest element
- Copy elements of data array into a new array "newData"
- Sort the array

⇒ **Array** - Collection of values
- fixed size

Primitive types - bytes (8) float (32)
Short (16) double (6
int (32) boolean (1
long (64) char

⇒ **JavaScript** - scripting language



Client side scripting
(JS) JavaScript

Server side scripting
(SSJS) Server Sid

Page No.
Date

HTML
<SCRIPT> </SCRIPT>

⇒ JDBC (Java Database Connectivity)
API - Collection of interface and classes
Interface - blueprint of class
Class - blueprint that defines (data) state and behaviour of entities (methods)

⇒ JSP - Java Server Pages
Similar to HTML.

⇒ Java Beans - We will map entity of data in JavaBeans.

Identifiers - is the name we provide.

Variable - is a memory location.

Rules for defining Variables -
* Case - Sensitive
Letters, digits, _, $
cannot have spaces.
Reserve words cannot be used.

Class has public interface and private implementation. method definition
data - instance var.

Page No.
Date

⇒ String class - defined in java language.
length()

```
Class A {
   public void print (String a)
}
```
↑ explicit parameter

```
A a = new A();
a.print("Hello");
```
↳ implicit parameter

⇒ Implicit parameter is the object using which the function is called.

⇒ Explicit parameter that we pass during fn call.

⇒ Number types

- Integral: byte, int, short, long
- Floating: float, double

Lab Ques: Implement the bank account class having instance variables Accno., balance (data) and methods float deposit(float a)
→ float withdraw (float a)
- int getAccountNo()
- void taxDeduction() (methods)

Page No.
Date

Implement class Bank having an array-list of bank accounts of type bank account and implement following methods

- addAccount() to Bank
- get total balance in Bank
- get acc no.s with min & max balance
- find a bank a/c given a/c no.
- Count the no. of a/c having atleast a specific balance.

⇒ import java.awt.*;
Rectangle - class
{(x,y, length(l), breadth(b)}
Rectangle r = new Rectangle(10,15,20,
x,y are the coordinates of top left most corner of rectangle.

⇒ Constructor must return reference to an object.

⇒ 2 types of Methods -

* Accessor methods
String x = "hello";
x.length();
do not change values of variables

* Mutator methods
Change values of variables (Obj)
r.translate(5,10) change x,y.

Page No.
Date

⇒ 2 types of methods -
parameterless
parameterized

⇒ Arrays - sequence of values.
double [] x = new double [5];
x.length;
↑ size

⇒ 2 common errors -
* Bound error - ArrayIndexOutOfBoun
* Uninitialized array.
double [] x;
x[0] // error as memory is not allocated.

⇒ Arraylists - predefined class
- Coluection of objects

~~type parameter~~
Arraylist <BankAccount> data =
class type

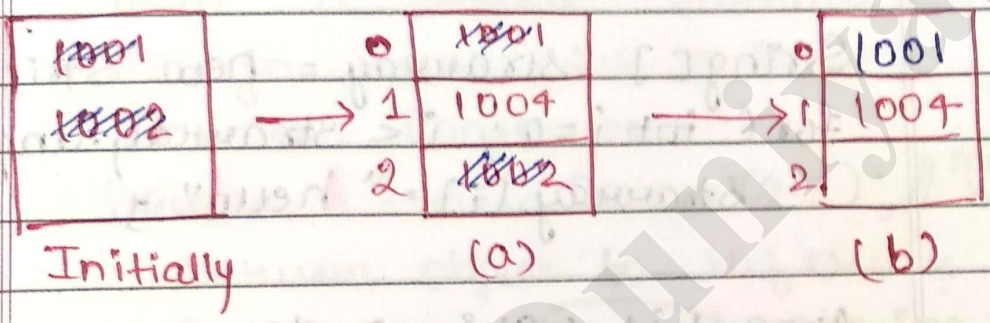
<BankAccount>
= new Arraylist(5);
↳ optional

- Arraylist is a dynamic array
- Can expand or compact on run time.
- to add element
data.add(new BankAccount (1001);

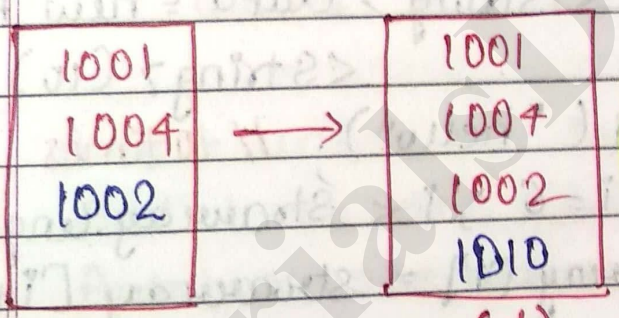
Page No.
Date

```
int n = data.size();
System.out.print("The array size is" + n);
```

```
add(int r, Object)
a) data.add(1, new BankAccount(1004));
b)   "    "   (new BankAccount(1001));
c)   "    "   (  "      "    (1002));
d) data.set(3, new BankAccount(1010));
```



Initially (a) (b)



(c) (d)

If we want to add an obj at loc 0 then all the elements at 0 or more will be shifted to next loc^n.

```
data.get(index)  // returns object
BankAccount x = data.get(0);
```

Error: Bounds error

```
BankAccount b = data.get(data.size());
```

String - length() Array - length
ArrayList - size()

Page No.
Date

```
* public class BankAccount {
    double balance;
    int accno;
    public BankAccount (int n)
    { accNo = n; }
```

Q: Create an array of 10 strings and an ArrayList of 10 strings. delete the 7th element from both of these.

Lab assignment

```
String[] strarray = new String[10];
for( int i=0; i < strarray.length ; i++
    strarray[i] = "hello";

ArrayList<String> data = new Arraylis
                         <String>(10);
data.add("hello"). // 10 times
```

To remove from array

```
for (int i = 6 ; i < strarray.length ; i++)
    strarray[i] = strarray[i+1];
```

To remove from AL → `data.remove(6);`

```
* Arraylist<String> names = new ArrayList(
names.add("A");          [A]      Allowed without str
names.add(0,"B");        [B|A]
names.add("C");          [B|A|C]
names.remove(1);         [B|C]
for(int i=0; i< names.size(); i++)
    S.O.P.L (names.get(i));
```

Page No.
Date

above versions 5

Wrapper classes (Auto boxing & auto-unboxing)

byte → Byte
char → Character
long → Long
float → Float
int → Integer
short → Short
double → Double
boolean → Boolean

eg:
```
double x = 2.5;
Double d = 2.5; // Auto-boxing
```

Steps in Autoboxing

* Create a double variable x.
* Assign value 2.5 to x.
* Invoke Constructor :-
  `Double d = new Double(2.5);` // manual boxing
* It returns object to reference d.
  `Double e = d + 1;` → unboxing

(i) Auto-unbox 'd' into a double value.
(ii) Add 1 to the value.
(iii) Autobox the value into new double.
(iv) Store a references to the newly created wrapper object 'e'.

* We cannot perform methods on primitive types.
We can perform operations on objs of wrapper class.
Thats why we convert primitive-wrapp

```
double x = d; // auto-unboxing
double x = d.doubleValue(); // manual unboxing
```

Page No.
Date

1/AUG/2019

Anonymous Array

```
public class MyClass{
public int meth ( int [] arr)
{
    return arr.length;
} } // class end.
Int  x = r.getWidth( ); // accessor method
MyClass ob = new MyClass( );
int x = ob.meth (new int[] {1,2,3,4});
```

Ques. Let there be a data ArrayList of type <Double> of size >0. How will you increment the element at index 0.

~~data[0] = data.get(0)~~ *

```
data.set ( 0 , ++data.get (0) );
```

⇒ For each loop or enhanced for loop

type parameter

```
ArrayList <Double> data = new
                ArrayList <Double> (2);

data.add ( 2.25); // autoboxing
data.add ( 3.25);
data.add ( 4.25);
data.add ( 5.25);
```

Page No.
Date

```
for ( Double x : data)
    S.O.P.L (x);
```

Or

```
for ( int i=0 ; i< data.size() ; i++)
    S.O.P.L ( data.get(i));
```

autounboxing

Ex: WAP to sum all the elements of an Arraylist of type <Double> using (i) for loop (ii) foreach loop

(i)

```
int Sum=0;
for ( i=0 ; i< data.size() ; i++)
    Sum += data.get(i) ;
```

(ii)

```
for ( Double x : data)
    Sum += x ;
```

⇒ Simple Algorithms for Arraylist / array

* Counting the matches
* finding a value
* finding minimum or maximu

1) To count the matches in Arrayli we check all the elements and count the matches until we reac the end of the Arraylist.

Page No. –
Date

Q. Count the no. of bankAccounts in accounts ArrayList (A·L) whose balance is atleast as much as a given threshold value.

```
public int count ( double threshold)
{  int count = 0;
   for( BankAccount x : accounts)
        if ( x. > = threshold)
            getBalance()
            count ++;
   return count;
}
```

(2) finding a value

Linear search through an A·L is defined as the inspection of each element one by one until we find a match or until we reach the end of the A·L.

Q. find whether there is a bank account with a particular account no.

```
public BankAccount find (int no)
{
   for (BankAccount x : accounts)
        if ( x.getAccountNO() = = no)
            return x;
} return ; or return null;
```

Page No. ____
Date | | | |

(3) Q: find the account with the largest balance and the lowest balance.

```
public BankAccount findMax()
{
double ~~int~~ ~~max~~ = accounts.get(0).getBalance(0);
~~BankAccount largest = accounts.~~

BankAccount max = accounts.get(0);

for (BankAccount x : accounts)
    if ( x.getBalance() > largest.getBalance)
        max = x;

return max;

}
```

Note - make findMin() also

⇒ 2-D arrays

* Tic Tac Toe game (O & X)

```
String[][] board = new String[3
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

Q. Create a class TicTacToe with a board named 2-D array & a constructor -

- which initialize the board with empty strings.
- set method that accepts the board position (i,j) and the player value. and sets the corresponding cell of the board
- toString() that prints the board values in the following format

```
|  -  x  -  |
|  -  0  -  |
|  x  -  -  |
```

- winner() which returns the player who has won the game.

```
class TicTacToe {
    String[][] board = new String[3][3];

    TicTacToe()
    {
        for(int i=0; i<3; i++)
            for(int j=0; j<3; j++)
                board[i][j]=" ";
    }
```

Page No.
Date

```
public void set(int i,int j,Strin
                                  player)
{   if( board[i][j] == " ")
    board[i][j] = player;
}

public void toString()
{
    for(int i=0;i<3;i++)
    {   S.O.P.("\n");

        for(int j=0;j<3;j++)
        S.O.P.("board[i][j]")

        S.O.P("" + "|");
    }
}
```

Create a class TicTacToeRunner which has the main() in which we request a user input interms of the cell coordinates where that user wants to play. Set the d initial player value as "X". Then kee on accepting cell coordinates and setting up the board until a player wants to quit or the board is full. until

⇒ Copying arrays

```
double [] data = { 3,4,5,6,7};

double[] newData = data;
```

(it will not copy the array but instead data & newData will refer to the loc^n.)

```
double [] newData =
(double[]) data.clone();
```

↳ it returns Object type

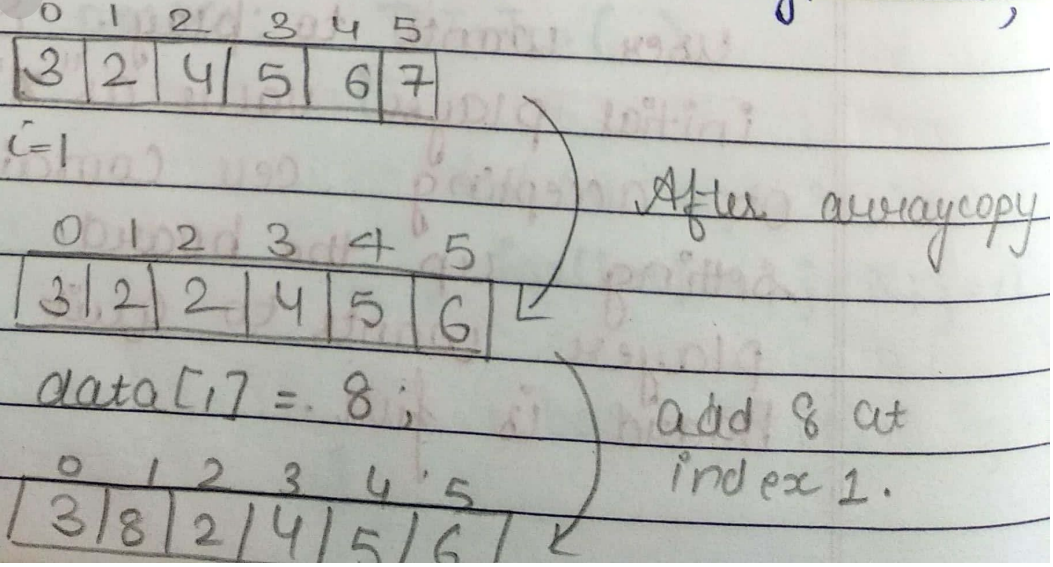
* to copy only few elements (*)

We use System.arraycopy (from, fromStart, to, toStart, count)

– This method has following purpose –

a) We use arraycopy to remove/add elements in the middle of an array.

eg – System.arraycopy(data, i, data, i+1, data.length-i-1);



Page No. _____
Date

Q. Use arraycopy() to create another array which is the double size of an given array & is its copy.

Q. Use appropriate method to insert/dele an element in an Arraylist. (Use A Metho

to remove element at i using arraycopy() -

```
System.arraycopy( arr, i+1, arr, i,
                  arr.length - i-
```

2019

JAVA SCRIPT

pre-requisite - HTML

* HTML
- to print some views or text on the browser.

* CSS
- for views or text styling.

* JS -
- scripting lang created by NetScape which is used to make a web page dynamic interms of content and presentation.
- It is an object oriented Scripting lang that provides event driven programming.

Event Driven Programming implies that JS recognises object based Events such as buttonClicked, mouseOver, Pageload etc. And it allows the code snippets associated with the events to execute when the event happen.

JS ~~sn~~ code is embedded within HTML program using <SCRIPT>... </SCRIPT>

Page No.
Date

- It is case-sensitive.
- JS supports many programming constructs such as condition checking (if else), switch case checking, loops like other programming langs.
- JS is client-side scripting lang which is often used for validating user i/p and error handling at the client side.
- JS is interpreted by the browser which has a JS engine.
- Browsers use HTTP to communicate with the web server and are designed to interpret and render HTML on a client Machine.
- The website development env. must provide a facility for validating the user i/p in such a way that the invalid user i/p does not attract repeat visits to the server. where the website is posted

```
<SCRIPT language = "JavaScript"
    or  type = "text/JavaScript">

</SCRIPT>
```

Netscape Communicator - browser which support JS. by default.

* Internet Explorer by default.

LIVE WIRE - product of NetScape that allows connectivity to database (Relational / nonRelational).

Create an HTML form to i/p the name, rollno., and batch of a student.

```
<HTML>
<HEAD>
<SCRIPT>
  Var name =

</SCRIPT>


<BODY>
  <form action = "file2.html" >
  <input type = "text" name = "Snam
  <input type = "number" name = "roll
  <input type = "text" name = "batch">
  <input type = "submit" value = "SEND">
  </form>
</BODY>
</HTML>
```

Page No.
Date

* JS is untyped that is it doesn't allow the datatype of var to be declared while creating a var.
* Same var may be used to hold different types of data at different times.

EXAMPLE -
```
var x = 10.25;
document.write ("<H2>" + x + "</H2>");
x = "Hello World";
```

⇒ PRIMITIVE TYPES

* Number → Integer
  → float
  → NaN

No.s are stored in the form of Octal, Decimal & HexaDecimal.

* Boolean → true → 1
  → false → 0

EXAMPLE -
```
x = 5;  y = true;
No. ← (z) = x + true;  // 6
```

* String - Sequence of 0 or more char enclosed within ' ' or " ".

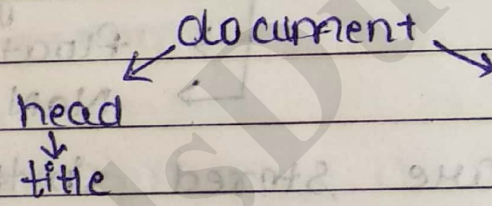
To include a ~~code~~ ~~corrector~~ quote character with the string we use (\).

Page No.
Date

* NULL - It contains the single value Null that identifies a non-existence reference or an empty value. It is used to prevent errors resulting from uninitialized var.

null → undefined
→ non-existent
→ Un-initialized.

⇒ DOM (Data Object Model)
↳ pattern/model representation of JS / HTML Pages.



⇒ Composite types
* Arrays
* Objects

⇒ Type Casting

var z = 25 + "200";
= 225 Number

var z = 25 + "Hello";
NaN.

"Hello25" = NaN or 25+"25 Hello" = 50

Page No.
Date

- Var in JS are loosely-typed & their types are defined implicitly based on the literal values that are assigned to it from time to time.

⇒ z = "Hello" + 25 ; // Hello25

* In an expression with different data types as operands the type of the first operand is considered for type casting.

* var name = prompt ("Enter name", "UserName");
or alert or Confirm
placholder

- prompt fn is a pre-defined JS fn which picks up the string from the user that is returned and assigned into a var.

- The script in the head section is interpreted first so the username is picked up first before anything is displayed in the client browser.

Page No.
Date

* variable defined in head tag are global.

⟹ Arrays → Topics → Syntax, Methods, Difference b/w Java & JS arrays

EXAMPLE - arr = new Array(1, 25.5, "ab", new Array(3,4));
↳ declaration + Assignment

arr = new Array(3); // sometimes ambiguous (length or ele.)
arr = new Array(); // Array size is 0. Use can dynamically insert ele.

⟹ Dense Array - It is an array that has been created with each of its ele being assigned a specific value. They are always declared & initialized at the same time.
So, listing all the elements value in the array declaration creates a dense array.

arr = new Array(val1, val2, ..., valn);

Page No.
Date

Array Methods

* join()
* Reverse()

EXAMPLE -

```
arr = new Array();
arr[0] = 2.5;
arr[5] = "hello";
```

- join() returns all elements combined as a single string. It takes only 1 argument i.e., a string to be used as the separator b/w each ele of the array by making the single string.

```
document.write(arr.join);
```

" 2.5, undefined, undefined, undefined, undefined, "hello" "

If the argument is omitted, join() uses comma-space as separator.

- reverse() reverses the elements of an array.

Property - arr.length is used to get the no. of ele in an array.

Page No.
Date

⇒ Property - defined as some aspect of the state of an obj.

⇒ Methods - used to read/modify the data contained in the objects properties.

⇒ Operators -

Arithmatic : + - / % * ++ , --

Logical : && || !

Comparison : == === != !== < <= > >=

Assignment : = += -= *= /= %=

Ternary : Condⁿ ? val1 : val2

String Concatenation : +

Special Character : delete operator (used to delete a property of an obj or an ele at an array index)
Eg. - delete arr[5];

new operator (to create instance of an obj)

Page No.
Date

void operator - (it does not return a value) It is typically used to return a URL with no value.

8/AUG/2019

Conditional Constructs

if - then - else, switch,
loops - for loop iterates on the basis of a condition or specific no. of times.

⇒ Function - blk of code that performs a specific task & often returns a value. It takes 0 or more parameters.

* Parameters provide a std technique through which ctl data is passed to a fn. Ctl data is used to ctl what a fn returns.

⇒ Built-in fn (a) & user defined fn's (b).

↓

explicit type fn's - parseInt(), parseFloat(), eval()

eval is used to convert a string exp. into a numeric value.

They are provided by the interpreter.

var total-value = eval ("10*5 + 11%3")

parseInt → String to int.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

~~System~~ document.writeln("Hello") → same as \n in Hello

Eg. - var String2num = parseInt("123"); Hello123 NaN

isNaN()
↳ true if given string is NaN else false.

parsefloat ⇒ String to float.

(b) function func (var1, var2 ...)
{ //body }
↳ without var keyword

* While declaring a fn we use keyword 'function' followed by a fn name and an optional list of parameters within parenthesis.
* fn can be declared anywhere in the HTML page within the script tag but is preferably placed in head tag. This allows the fn to be parsed before being invoked.
* Parsing is the process by which JS interpreter evaluates each line of script code & converts it into a pseudo-compiled byte code before attempting to execute it. At this time Syntax errors & other programming mistakes that would prevent the script ~~code~~ from running are trapped & reported.

Page No.
Date

Ques.

```
<BODY>
  <FORM  name = "myform"
         action = "Second.html"
         method = "Get"
         OnSubmit = "return validate()">
    Enter Name  <input type="text"
    name="username"  placeholder = "abc" >
    <input type="Submit"  value="Send">
</FORM>
</BODY>

<HTML>
<HEAD> <SCRIPT>
function validate()
{  var name = document.myform
     .username.value ;
   if (name == null || name == "")
   { alert("      ");
     return false;
   }
}
</SCRIPT> </HEAD> </HTML>
```

by default it returns true

⟹ Difference b/w Global var & local var

* Parameters of the fn as well as the va declared within the fn with var keyword. ~ local var.
* within the fn you can declare glo var as. y=20; (without var key

Page No.
Date

* if the local var declare with same name as global var then within the fn that var name will refer to the local var.

⟹ Dialog boxes → prompt, alert, confirm

* Confirm
Eg. var x = confirm("Do you want to exit?");
// it will be with 2 btns. OK & Cancel
// OK returns true & Cancel returns
// false).

⟹ Advantages Of JS. (read from book)

* interpreted lang
* High Performance
* Event based
* Platform & ~~architect~~ architectural neutr
* Minimal Synta
* User driven
* Easy to debug

Theory Asg 2. (1) Global & local var. How does the use of datatypes in context of var defer in Java & JS.

(2) WAJS code to prompt the user to enter n numbers & print the count of (-) nos, (+) nos and 0's.

Web Asg 3 (3) WA JS code to display current date time in browser which should keep

Show Dynamic time

show date-time()
{var x = new Date(); } wrong
body ← SetTimeout(show date-time(), 1000);
in while loop (in millisec)

updating itself every second.

(4) Explain the f^ns of action & method attrs in the form tag of HTML. 3marks.

(5) What is NaN in JS? Explain.

(6) Write a JS code to implement Dense Arys.

(7) WA JS code which will Greet user according to current time.

(8) Explain the following JS methods with egs. eval(), parseInt(), join(), substring(), alert()

(9) Lab. WA JS code blk with the following validations for username & Password.

a) if the name/Password is not entered, display an error msg showing "You forgot one of the required fields. Try Again"

b) incase the fields entered do not match the hardcoded values, display an error msg showing enter valid username & password

c) if the values entered match display the msg "Welcome" username.

Lab 3. Implement Stack class.
static stack = array
dynamic stack = arraylist

(10) WA JS code blk which checks the contents entered in a form's text ele. If the text entered is in the lower convert to uppercase

(1) Fill in the blanks - 13
True & False — 9 } Pg 148

UNIT-2

CHAPTER-9

Page No.
Date

14/AUG/19

Document Object Model (DOM)

HTML elements are mapped into DOM Object before being rendered (painted) on the web browser.

DOM objects forms an Instance hierarchy :-

↦ Navigator
  ⟼ Browser's Window
    ↦ Document
      ⟼ Anchors (<a>), Links (<), forms

→ DOM is used to allow a browser to recognise html Obj's even after rendering them. To make a web-page interactive it is necessary that browser continues to recognise the html Obj's even after they are rendered in the browser window. In a DOM the collection of HTML page elements have a descending relationship with each other. This hierarchy is called DOM or object hierarchy.

* Navigator is at the topmost level in this hierarchy.

* JavaScript Obj hierarchy or instance hierarchy is mapped to the DOM

Page No.
Date

which is further mapped to the html ele.
JS Instance hierarchy → DOM → HTML elements.

* JS is ∴ capable of recognising each ele in a webpage & all these ele's are ~~mount~~ bound to "~~bound~~ DOM".
* Each html ele gets registered only after it is assembled in the memory of the browser before getting rendered in the browser window.
* If a document for eg does not have an anchor ele the Anchor's Obj will exist but it will be empty.

Plug-ins - a software which is added to DOM or any other to provide additional feature.
ug-ins like Applets & Images are recognised by JS enabled browsers in addition to DOM.
Eg. <HEAD> ... </HEAD> <BODY> ... </BODY>

JSSS

Presentation styles, headings, body text etc. are not part of DOM & ∴ are not recognised by JS. But they are recognised by JS only if JS Assisted StyleSheets. only JSSS is a webpage.
JSSS is embedded usually in b/w the <HEAD> ... </HEAD>

Page No.
Date

JSSS uses JS syntax to ctrl the presentation style of a document.

↳ Document
↳ Tags
↳ P; DIV, SPAN, H1 to H6
↳ Classes
↳ Tag Names
↳ IDs

EXTENDED DOM Obj Hierarchy

The DOM recognised by JS gets extended by embedding JSSS in a webpage. This helps the developer to access every element of a webpage whether or not that ele appears on the page when rendered.

General syntax to access -

* Property - ObjName.PropertyName

* Methods - ObjName.methodName()

* Events - allow JS code snippets to be connected to the obj being mapped to corresponding event handler. The occurence of the event leads to this execution of

Page No.
Date

the code snippet. This is the traditional object driven, event driven, code execution Model.

⇒ Object based & Object Oriented

JS is Object based not object oriented bez. it doesn't fully support basic OOP capabilities like inheritance, polymorphism, encapsulation, abstraction. It is called Object based bez. it can recognise pre-defined browser Obj's & Server Obj's & it can use the behaviours of these objs through their properties & methods.

	JS ObjName	Purpose
1.	navigator	– use to access the info about the browser in which the current JS is executing.
2.	window	– use to access browser window or a frame within the window.
3.	document	– use to access the doc currently loaded into the window.
4.	location	– used to represent a URL. It can be used to create a URL Obj or modify an existing URL etc.
5.	history	– it stores a history of all the URLs accessed within the window.
6.	event	used to access info about the occurence of an event.

Page No.
Date

7. EVENT – this obj provides constants that are used to identify events.

8. Screen – used to access info about the size & color depth of a client comp. screen in which the browser is running.

* Any document can contain many html objs such as hyperlinks, images, image Maps, frames, applets, multimedia obj, forms and form ele" etc.

* The browser creates an array in memory per html obj in the doc thus registering each of these html objs.

* If these html obj accutually exists within the html page then these arrays hold indexed ele's which pt. to a context area in memory where the html objs are located. Otherwise the array will exist but it will be empty.

* If there are multiple similar html ob in a doc then the array will have multiple indexed ele's.

* JS provide access to arrays & its ele

Page No.
Date

The values held in the array ele's pt to the context area in memory where the html obj's are stored. These html obj's can be accessed using properties and methods.
Eg- forms[i].name
So, the functionality of a html obj is controlled dynamically while an html page is running in the browser.

⇒ Arrayname and purpose

1. Images - it is used to access an img or array of imgs embedded in the html page.
2. Links - used to access a src anchor of a hypertext link.
3. Area - used to access an area within a client side image map.
4. Frame/Frames - used to access a html frame or an array of all frames in a window.
5. Anchor/Anchors - used to access the target of a hyperlink or all the anchors in the doc ~~hyperlink~~
6. Applet/Applets - used to access the java applet or all the Applet objs in the doc.
7. Embed/Embeds - used to access embedded obj or all the embedded obj.s in the doc.

Page No.
Date

8. Mime Type / MimeTypes - used to access info about a multimedia type supported by the browser or all the multimedia types.

9. Plug-in / Plug-ins - used to access info about particular browser plug-in or all the plug-ins

10. form / forms - used to access an html form or all the forms in the doc.

11. Form Elements - used to access the following form eles - text, textarea, radio, checkbox, button, submit, reset, select, option, password, hidden, file upload.

⟹ EVENT HANDLERS -

(1) Interative - user is interacting with an HTML ele. Eg. onClick.

(2) Non-interactive no user interaction is required. to be invoked Eg. onload

(i) Depends on the user interation with an html page. Eg. onclick.

Read from book. - Eg.s of eventhandlers.

type ... control to file ... is specified in action ... of form.

Page No.
Date

Ques. L-A-4. Create a webpage using 2 img file which switch b/w one another as the mouse ptr. moves over the imgs. Use the onMouseOver and onMouseOut event handlers.

Ques. T-A-3. 1. What is a DOM model in JS ? (5 marks)

~~2. Explain the follo~~ 2. Write JS code to validate phone no.

3. Define a user-defined nested Obj (at least) with properties - make, model, year and owner where owner is a nested Obj with properties name, age & gender.

(X) Write the JS code to greet a user acc. to current time.

CHAPTER-10

Forms used by a Website

```
<FORM name="survey" method="get" action=
"Second.html">
<input type="text" name="firstName"
size="25" value="Enter a name"
selected>
<input type="radio" name="status"
value="fresher"/>
```

To check whether the radio is checked or not: ns[i].element[i]. Yes </BR>

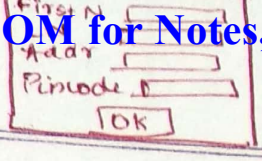
```
<input type="radio" name="status"
value="experienced"/> NO<BR>
<input type="button" name="bt" value=
"Press OK" onClick="count()">
```

(Ctrl) button to the same file.

Q: Set the value of checkbox and radiobuttons programmatically using JS code.

```
<form name="myform" onload="SetButtons">
    <input type="radio" name="gender"
        value="male"/> Male <br>
    <input type="radio" name="gender"
        value="female"/> Female <br>
    <input type="checkbox" name="exp" value=
"yes"/> Yes
    <BR>
    <input type="button" name="btn"
        value="Check">
</FORM>
function SetButton()
{
    document.forms[0].gender[0].checked
        = true;
}
```

Q: Create an HTML form that has a no. of textboxes when the form run user fill the form. Write the JS code that verifies that all the text boxes have been filled. If a textbox is left empty then popup an alert indicating which textbox have been left empty. When the 'OK' btn of the alert box is clicked on set the focus to the specific textbox. If all the TB are filled display a thankyou alert.



Page No.
Date

```
<HTML>
<HEAD>
<SCRIPT>
    function onValidate()
    {
      var fn = document.forms[0].fn.value;
      var ln = "" "" .ln.value;
      var addr = "" "" .ad.value;
      var pin = "" "" .pin.value;

    if (fn=="")
    alert("First Name field is empty")
    if (ln=="")
    alert("Last Name is empty")
    if (

    for(var i=0; i<elements.length-1;
                                  i++)
    {
      if (document.forms[0].element[i].value
                                  =="")
      {document.forms[0].
      elements[i].focus();
        alert("elements[i]"+"is empty");
      }
    }
    }
  </SCRIPT>
</HEAD>
<BODY>
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

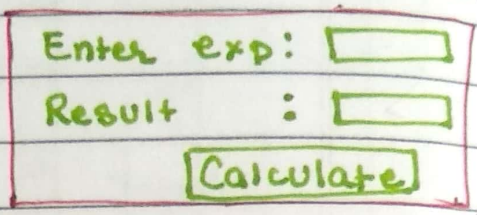
```
<FORM onSubmit="onValidate()">
First Name:
<input name="fn" type="text">
Last Name:
<input name="ln" type="text">
Address:
<input name="ad" type="text">
PinCode:
<input name="pin" type="number">
<input type="button" value="VALIDATE">
</FORM>
</BODY>
</HTML>
```

FORM ELEMENTS

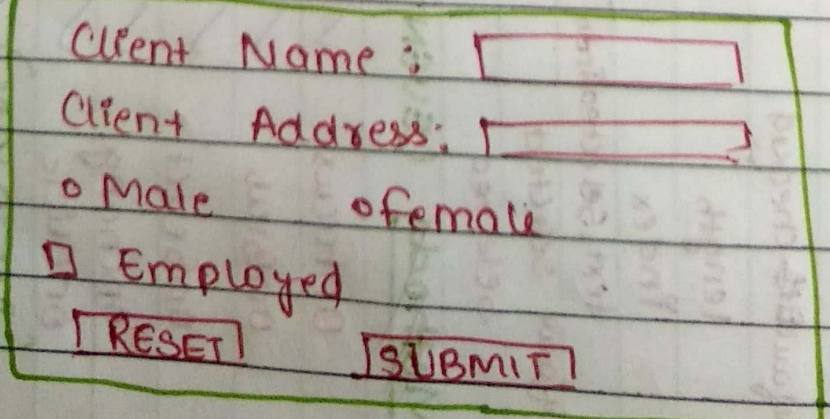
Text textarea password button. radio CB

Page No.
Date

Ques: Develop an HTML page that accepts a mathematical expression, evaluates a exp and then displays the result of the evaluation.




```
<HEAD>
<SCRIPT>
   function calculate()
   { document.form1.result.value = eval(document.
                    form1.entry.value);
   }
</SCRIPT>
</HEAD>
<BODY>
     <form name="form1">
      Enter an expression:
      <input name="entry" value=""/> <BR/>
      <input type="button" name="bt" onclick=
                "calculate()"> <BR/>
      Result:
      <input type="text" name="Result"
                onfocus="this.blur()">
     </form>
</BODY>
```



Page No.
Date

lab.

Q.2 Make a webpage which uses two textfields and one checkbox. The first TB obj accepts a numeric value. Depending on the checked/unchecked state of the CB., the second TB obj displays the following -

a) CB is not checked then display the x2 of the no..

b) CB is checked disply the sq. of the no.

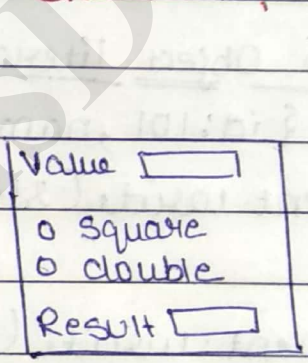
If the second TB obj is loaded with a numeric value, depending upon the checked/unchecked state of the CB, the first TB obj displays the following -

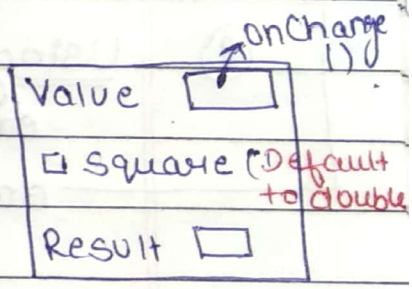
c) Checkbox (CB) is not checked then display half the no. entered.

d) If CB is checked, sqrt of the no.

lab.

Q.3 An html form look like.-





⟹

```
onClick = "calculate(this.form1);   // obj.
// if we are using calculate() in
// a single form only then we can
// also pass form name.
function calculate(f1) { }
```

Date

textarea

```
<textarea name="add" value="ABC"
          row="6" cols="20" onFocus="this.blur()">
...
</textarea>
```

28/AUG/2019

→ Object-based

⟹ Ways to create an object in JS

① Using Constructor

```
function Student(srollno, name, marks)
{  this.srollno = srollno;
   this.name = name;
   this.marks = marks;
}
```

Eg:→ `var s1 = new Student(11,"Neha",350);`

② Using an Object literal

```
emp = {id:101, name="Vikas", salary:50000};
document.write(s1.srollno + " " + s1.name+...);

document.write(emp.id + " " + emp.name +
                " " + ...);
```

③ By creating an instance of an object

```
var emp = new Object();
empid = 11;
emp.name = "abc";
```

⟹ Built in Objects

String, Date, Math

Page No.
Date

var s = "abc";

or var s = new String();
s = "abc";

or var s = new String("abc");

* length is a property of String.

* Methods in String Object

(i) s.big() ⇒ surrounds the string with the html's big tags.

(ii) s.blink() ⇒ surrounds the str with the html's blink tags.

(iii) s.bold();

(iv) s.charAt(index);

(v) s.italics();

(vi) s.tolowercase();

(vii) s.touppercase();

(viii) s.substring(start, end)
index-1

* Math Objects

Properties - E ⇒ (Euler's constt)
LN10 (Natural log base 10)
LN2 (Base 2 log)
PI (π)

Methods

(i) abs() (iv) sin (vii) sqrt
(ii) ceil() (v) cos (viii) random
Blw 0 & 1.
(iii) floor() (vi) tan (ix) pow

Date

* Date Object

~~Properties~~ Components
year, month, day, hours, mins, secs, milisecs

Methods

d.setFullYear	setTime getTime
d.getFullYear	↳ sets time based
setDate getDate	on the arg. repre
setMonth getMonth	senting no. of mil
setHours getHours	second since
	1 Jan 1970 00:00:00
	:00

⟹ Objects within Objects (user-defined)

```
function Student (rollno, name, marks)
{    this.rollno = rollno;
     this.name = name;
     this.marks = marks;
}

function Marks (eng, sci, maths)
{    this.eng = eng;
     this.sci = sci;
     this.maths = maths;
}

m1=new Marks(100,90,95);
s1= new Student(101, "Vivian", m1);
```

Page No.
Date

→ Window Object

* represents a window within a browser. The Obj is automatically created by the browser. Window is not a JS obj it is an object of the browser.

* The methods of the Window Obj –

open() - it creates/opens a new window.
close() - it closes the current window.
prompt() - it displays a dialog box to get the input from the user.
alert() - displays an alert box containing a msg & a "OK" btn.
confirm() - displays a confirm dialog box containing a msg "OK" & "Cancel" btn.
setTimeout() - It is an method of System and Window both. It performs an action such as calling a fn or evaluating an expression after a specified time in milliseconds.

* Signature of open()

windowsopen(url, name, specification, replace)

url – it will open a new browser window. with the specified url of the page. It is optional (" ") if no url is specified a new window opens with about:blank.

Page No.
Date

name ⇒ optional
name of the page/window
following values are supported in the name arg. blank, parent, self, top and name.

blank - url is loaded into a new window

parent - url is loaded in to parent frame.

```
<frameset '50%,50%'>
  <frame>...>
</frameset>
```

self - url replaces the current page

top - replaces any frame sets that will be loaded.

name - it will simply apply a name to the page.

Specification it is (,) separated list of items which supports following values.

Name	Value	Description
channelmode	Yes/No/1/0	It tells whether to disp a windown in theatre mode or not.

Page No.
Date

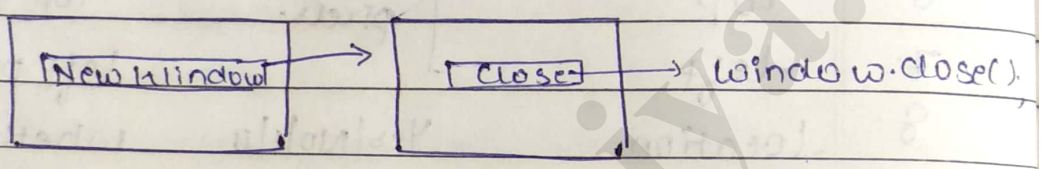
2.	directories	Yes/no/1/0	it specifies whether dir. btns should be added or not.
3.	fullscreen (Taskbar is not displayed.)	Yes/no/1/0	whether or not to display browser in full screen mode.
4.	height	values in pixels.	height of the window
5.	width		width " " "
6.	top		top pos " " "
7.	left		left pos of the window
8.	location.	Yes/No/1/0	whether or not to display the addr. field.
9.	menubar	Yes/no/1/0	whether or not to display the menubar
10.	resizable	Yes/no/1/0	window is resizable or not.
11.	scrollbars	Yes/no/1/0	display or not
12.	status	Yes/no/1/0	disp titlebar or no
13.	toolbar	Yes/no/1/0.	" tool " " "

replace - specifies whether the url creates a new entry or repla
the current entry in the history li
When replace is "true", it will replace
the history's current entry of the pa
with the current name.

when it is "false" it creates the new entry.

Q. Open a new window when a btn is clicked. The new window opened is closed by placing a btn in the new window & writing JS code in the btn's onclick event.

```
var newwindow = window.open( );
```



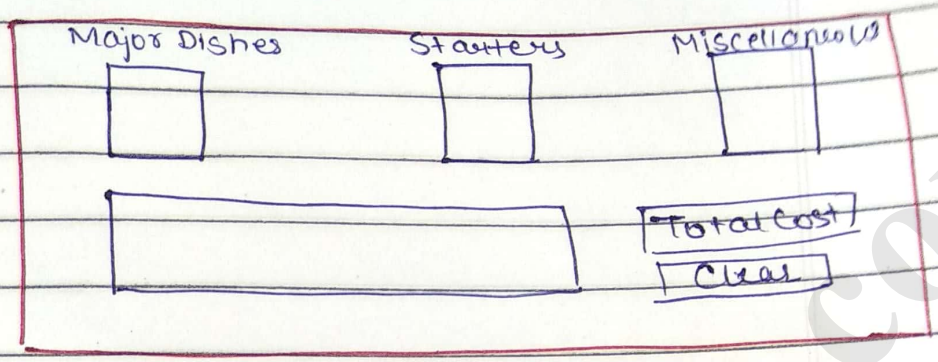

```
newwindow.document.write("<HTML>
                          </HTML>");
```

Q. Make an html form with two multiple choices & one single choice list. The first multiple choice list disp the major dishes available. Second multiple choice list disp the starters & single choice list disp miscellaneous items that include icecream, milkshakes & soft drinks.

The selected items must be displayed along with its cost in a textarea. On clicking the total cost btn the total cost is

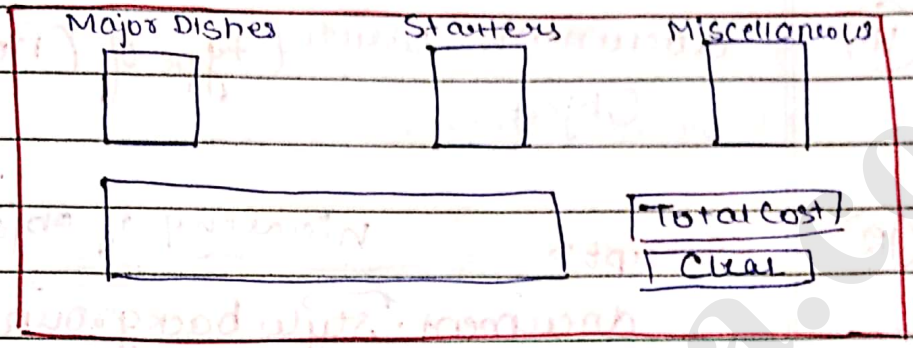
Page No.
Date

computed & displayed at the end in the textarea. The 'clear' btn is provided to clear the textarea.



Page No. _____
Date

computed & displayed at the end in the textarea. The 'clear' btn is provided to clear the textarea.



29/AUG/19

Q: (1) var a = "5" + 5; //55

(2) arr = new Array(5); ambiguous but its correct. It will consider 5 as size
document.write(arr.join(","));

(3) <HTML><BODY><form name="myform">
<input type="radio" name="agegroup"
value="<18"> (<18) <input type="radio"
name="agegroup" value=">18"> (>18) eg: <18 or >18
</form></BODY></HTML> //error \>18

<18 >18

(4) var a = eval("5") + 5; // 10

(5) employee = {20, "Mohan", 20000};
document.Write(employee) // this is also allowed
//error employee = new Array(20, "Mohan", 2000);
document.write(employee.join());

(6) var x = 2 + "3" + 5; // 235
document.write(x); // NaN

(7) document.write(parseFloat(10+5)); // 15.0

(8) alert("Welcome", "Guest"); //error
alert("Welcome" + "Guest");

Page No.
Date

(9) x = 10; y = 9; error.

(10) document.write("356" === 356); false.
// false

(11) document.write(typeof(new Date()));
Object.

(12) <Script>
document.style.background.color = "red";
</Script> // color of background change
(or) document.body.style.background.color = "red";
hierarchy of objects.

(13) a = 400
document.write(a.sqrt()); // error
Math.sqrt(a); // 20.

(14) sample = "Hello, World".
alert(sample.toUppercase() + sample.substring(0,5) + "!");
O/P:- HELLO, WORLDHello!
error - toUppercase()

(15) a = "frenchfries → 80";
b = a.charAt(">"); // charAt (valid index)
c = a.substring(b+1, b+3);
document.write(parseInt(c+10));

a = "frenchfries → 80";
b = a.indexOf(">"); // 13
c = a.substring(b+1, b+3); // 14 & 16
document.write(parseInt(c+10));
// 8010

Page No. _____
Date

⇒ history - it is an obj

Go to prev page { <input type="button" name="back" onclick="history.back()> Back

Goto next page { <input type="button" name="forward" onclick = history.forward()> Forward

frameset - frame

A

Ch-1 ⑧ → Example- 11, 12, 13 & Hands On Exercise1

Que. A webpage contains 3 frame. Top frame contains entry fields for background color, text color, link color, & button to enable the users to test their color combinations. When the user presses the btn the script loads a simple doc using the specified colors in the entry fields into the lower frame.

⇒ frame set - It is a tag which is used to divide a window into multiple frames.

The <frame>...</frame> defines a specific window where we can load another webpage. for this we use src attr.

* NOTE - frameset tags are outside the head and body tag. Body section is not mandatory here.

```
<frameset rows="50%,50%" cols="50%,50%">
    <frame src="abc.html" />
    <frame name="output" src="def.html" />
    <frame src="ghi.html" />
    <frame src="jkl.html" />
</frameset> </html>.
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

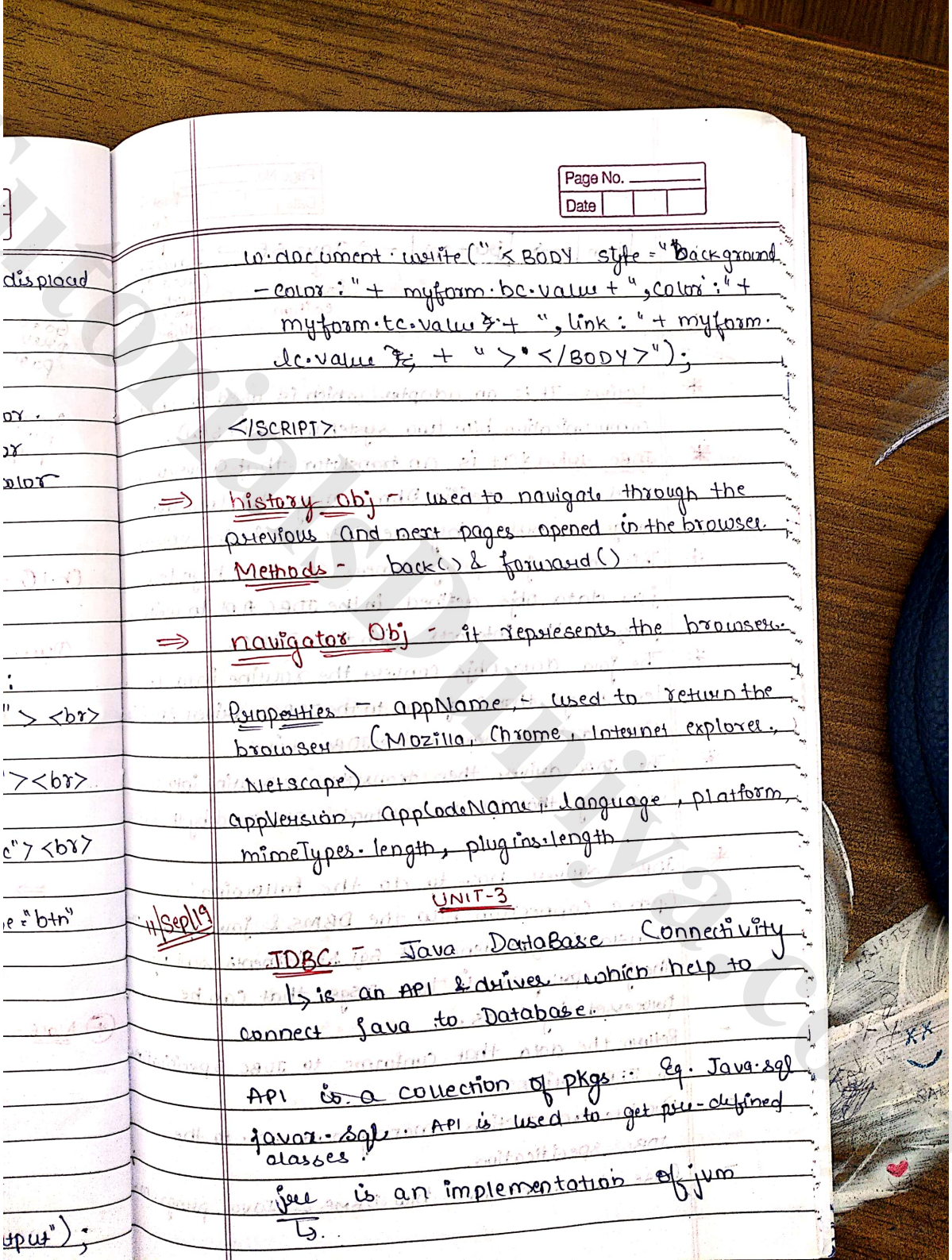
<noframes> ... </noframes> allows alternate content to be displayed where frames are not supported.
Used immediately after </frameset>

background color - document.bgcolor
text color - document.fgcolor
link color - document.linkcolor

Ques Ans: a1.html

```
<HTML>
<BODY>
  <form> name="myframe"
    Enter Background Color :
    <input type="text" name="bc"> <br>
    Enter Text Color :
    <input type="text" name="tc"><br>
    Enter Link Color :
    <input type="text" name="lc"> <br>
    <br>
    <input type="submit" name="btn"
      value="SUBMIT"
      onclick="display()">
  </form>
</BODY>

<HEAD>
  <SCRIPT>
    function display() {
    w = window.open("","output");
```



Page No.
Date

w.document.write("<BODY style="background-color:" + myform.bc.value + ",color:" + myform.tc.value + ",link:" + myform.lc.value + ">");

</SCRIPT>

⇒ history obj - used to navigate through the previous and next pages opened in the browser.
Methods - back() & forward()

⇒ navigator Obj - it represents the browser.
Properties - appName - used to return the browser (Mozilla, Chrome, Internet explorer, Netscape)
appVersion, appCodeName, language, platform, mimeTypes.length, plugins.length

11/Sept/19

UNIT-3

JDBC - Java DataBase Connectivity
↳ is an API & drives which help to connect Java to Database.

API is a collection of pkgs. Eg. Java.sql, javax.sql. API is used to get pre-defined classes.

Jvm is an implementation of jvm
↳

Page No.
Date

Not in course

CODE

jvm → jre + api → Java SE → Java EE (used for web app's)
↓
Java std edition
Java enterprise edition

Driver - It is an adapter which is used to enable communication b/w two systems (H/w or S/w). (1)

JDBC driver - It is an translator that converts low level vendor specific Dbms msgs to low level msgs understood by the JDBC API & vice-versa. (2)
This means java programmers can use high level java data objs defined in the JDBC A-PI to write a program that interacts with the DBMS. (3)
The java data objs convert the routine into low level msgs that conform to the JDBC driver specification and send them to the JDBC driver.
The JDBC driver then translates the routine into low level msgs that are understood & processed by the DBMS.

JDBC drivers have to do the following:- (4)
Open a connection b/w the DBMS & java program
Translate equivalence of SQL statements sent by the java program in two msgs that can be processed by DBMS.
Return the data that conforms to JDBC specification to JDBC drivers.
Provide transaction management acc. to the JDBC specification.
Close the connection b/w DBMS & Java program

Visit TUTORIALSDUNIYA.COM for Notes, Tutorials, Programs, Question Papers

CODE

Page No. _____
Date

```
package examples
import java.sql.*;
public class MyExample {
    public static void main(String args[])
    { try { Class.forName("com.mysql.jdbc.Driver");
```

① predefined class → load driver & register JDBC

```
Connection con. = DriverManager.getConnection(url,
```

② interface ↑

```
                  String userid, String password)
```

optional ("root", "password")

```
url :- "jdbc:mysql::3306/college"
```

↳ dbname

```
Statement stmt = con.createStatement();
```

③ interface ↑

// createStatement() returns the obj of class that implements Statement interface. similarly getConnection returns an obj of class that implements Connection interface.

```
ResultSet rs = stmt.executeQuery("select *
                from Student;");
```

④ class ↑

// executeUpdate() is for DDL
// executeQuery is for DML.

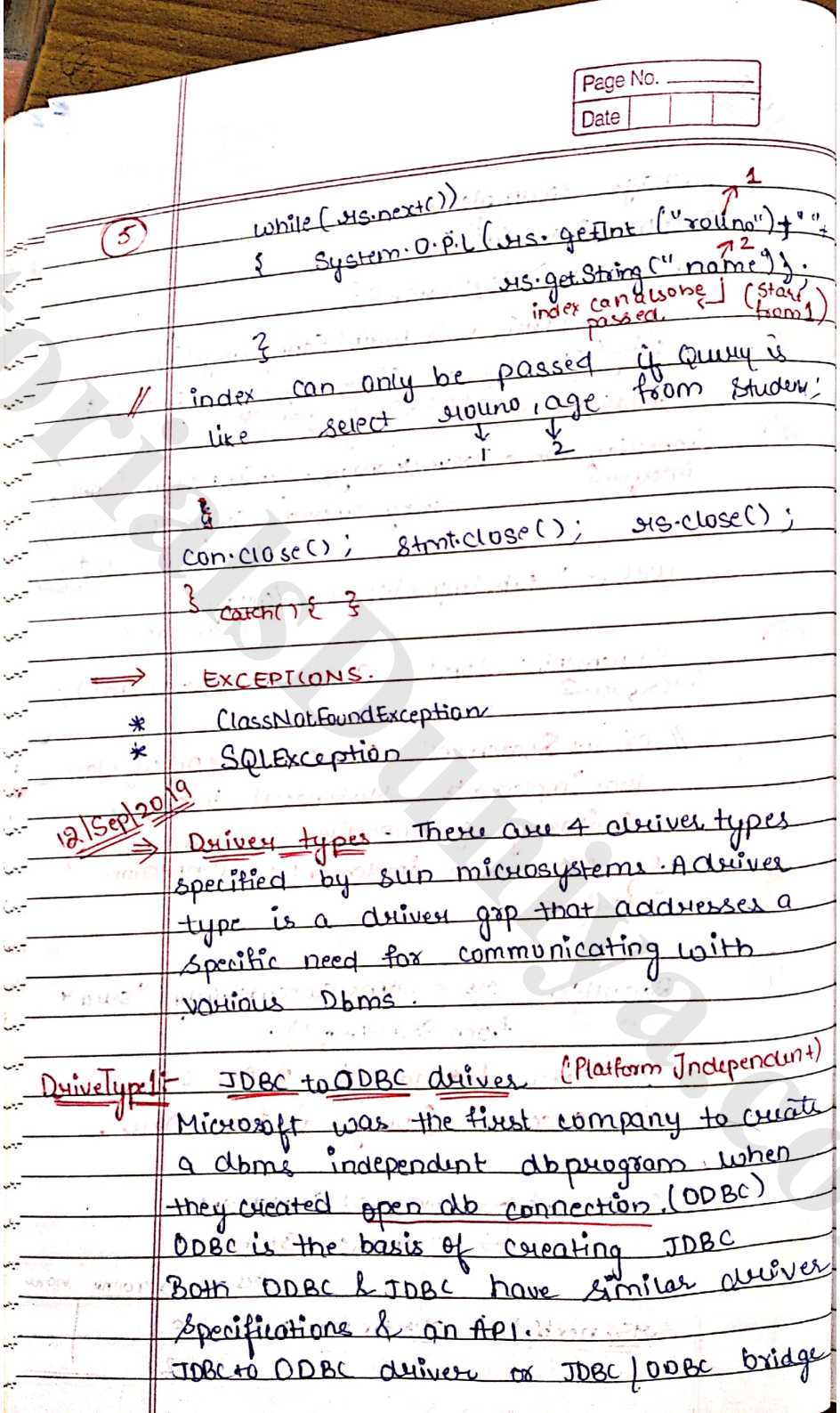
Statement → preparedStatement
↳ CallableStatement.

rs → rollno | name

0
1
2

rs.next() // pts to next row

Visit TUTORIALSDUNIYA.COM for Notes, Tutorials, Programs, Question Papers



Page No.
Date

(5)

```
while(rs.next())
{ System.o.p.l(rs.getInt("rollno")+" "+
                rs.getString("name"));
}
```

1 — index can also be passed (start from 1)
2

// index can only be passed if Query is like select rollno, age from Student;
1 2

```
}
con.close(); stmt.close(); rs.close();
} catch(){ }
```

⟹ EXCEPTIONS.

* ClassNotFoundException
* SQLException

12/Sept/2019

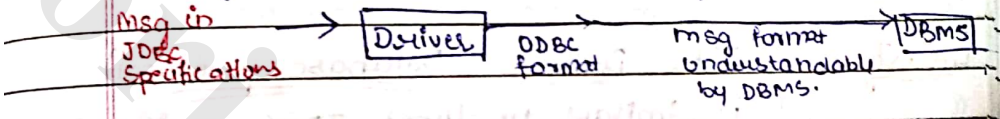
⇒ Driver types - There are 4 driver types specified by Sun microsystems. A driver type is a driver grp that addresses a specific need for communicating with various Dbms.

DriveType1:- JDBC to ODBC driver (Platform Independant)
Microsoft was the first company to create a dbms independant db program when they created open db connection (ODBC)
ODBC is the basis of creating JDBC
Both ODBC & JDBC have similar driver specifications & an API.
JDBC to ODBC driver or JDBC/ODBC bridge

is used to translate the dbms calls b/w JDBC specifications and ODBC specifications.

Step1 :- This driver receives msgs from java program in the format that conforms to JDBC specifications/rules

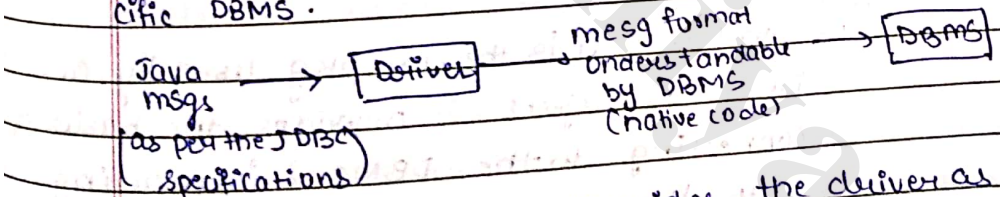
Step2 :- These msgs are then translated by the driver into ODBC msg format which is then translated into the msg format understandable by dbms.



Avoid using JDBC/ODBC bridge in mission critical apps bcz the extra translation might negatively impact performance.

DriverType2 JAVA/native Code Driver

Step1 : This driver uses java classes to generate platform Specific code i.e., the code understandable by the Specific DBMS.



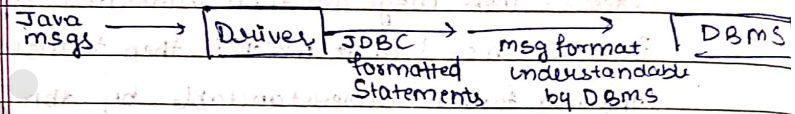
Step2 The DBMS manufacturer provides the driver as well as the API classes so that java msgs can be converted into platform Specific code.

Disadv :- loss of portability of code which leads to non-linkability of the API & driver with another manufacturers DBMS.

DriverType3 JDBC driver / Java Protocol Driver

Page No.
Date

most commonly used driver which converts sql queries into JDBC formatted statements. These statements are translated into the format required by DBMS.



DriverType4: JDBC Driver (Database protocol)
It is similar to type3 JDBC driver except that the sql queries are directly translated into the format required by DBMS: i.e., they do not need to be translated into JDBC format statements. This is the fastest way to communicate sql queries to the DBMS.

⟹ JDBC Pkgs - The JDBC API contains 2 pkg

① java.sql - It is the core pkg used by core java data objects. It provides the basics of connecting to the DBMS & interacting with the data stored in DBMS.

② javax.sql - This pkg extends java.sql It is in JavaEE. It has data objs that interact with Java Naming Directory Interface (JNDI) and it has data objs that manage connection ~~to link~~ pooling & other advance JDBC features.

Page No.
Date

⇒ Steps for JDBC Connection Process

* Loading JDBC driver.

java.lang.Class pkg. Class.forName() is used to load the JDBC driver. The name of the driver is passed as the parameter to this method.

foreg :- A developer wants to work offline & create a Java Program that interacts with Microsoft Access on his comp. for this we must write a routine to first to load the JDBC/ODBC bridge called "sun.jdbc.odbc.JdbcOdbcDriver"

* Connect to the DBMS - Once the driver is loaded the java program must connect to the DBMS using DriverManager.getConnection() In java.sql.

java.sql.DriverManager is the highest class in java.sql pkg hierarchy and manages the driver's information.

URL of the db, Username Id & Password are passed to the getConnection(). URL is a String which contains driver name & db name. This method returns a Connection interface used to refer to the db.

java.sql.Connection interface manages the communication b/w the driver & java program. It sends the statements to DBMS for processing.

URL :- "jdbc:odbc:college"

* Create and execute an SQL statement

The next step is to send an SQL query to DBMS for processing. an sql query contains a series of sql commands that direct the dbms to do something. for eg.- return the rows of data to the java program.

Connection con;

con.createStatement() is used to create a Statement Obj. This obj is then used to execute a query & return a ResultSet obj that contains the response from DBMS. This ResultSet obj contains one or more rows of info most of the times.

Query is same as a String which is passed in the Statement obj's executeQuery().

After receiving the ResultSet, close() is called to terminate the statement. (stmt.close());

Class.forName("com.mysql.jdbc.Driver");

Connection con = DriverManager.getConnection ("jdbc:mysql::330 "root", "password");

~~com.mysql.jdbc.Driver~~

Statement s = con.createStatement();

Resultset rs = s.executeQuery(—);

* Process data returned by DBMS.

java.sql.Resultset obj is assigned the result received from DBMS, after

the query is processed, the ResultSet contains methods used to interact with data returned by DBMS.

Eg: a customers table has fields firstName & lastName

```
ResultSet rs = stmt.executeQuery("select firstName, lastName from customers;");
while(rs.next())
{ S.O.P.L("firstName:" + rs.getString(1) + ", Last Name :" + rs.getString(2));
}
// SQLException when there is error in query
```

The next() method returns true, if atleast one datarow is present in the ResultSet

getString() returns the copy of a column value in current row of ResultSet into a String Obj. The name of the col in ResultSet whose content is to be copied is passed to the getString().

Instead of passing the colname to the getString() we can also pass the Col no., we use colno only when we specify the colname in select stmt. Otherwise we ~~use~~ cannot be sure of the order in which the cols appear in the ResultSet, since a table may be reorganized since it was created & col might be rearranged.

rs.previous()
rs.first() # last row
rs.last()

Page No.
Date

The conditional argument of while loop (rs.next()) returns false when the ptr is at the end of the ResultSet.

* Terminate the Connection to the DBMS.
close() of Connection Obj is used to terminate the connection to the DBMS. It throws an exception if a problem occurs in disengaging with the DBMS.
Eg. con.close();

⟹ DATABASE CONNECTION

⇒ J2EE component
↳ 1. Client component 2. web component

* J2EE components includes java programs for the GUI, for processing user request, fetching the data etc. There are different types of Java EE components such as client end component, web component, business component etc.
(All standalone Appln are J2EE client component).

web component - JSP, Servlet, HTML, SSJS etc.
Business component - Enterprise Java Beans (EJBs)

→ J2EE containers provides the interaction of J2EE components with the underlying functionality & the platforms. It includes different types of servers such as db servers (eg. MySQL Server) & Web Servers (Apache Tomcat), Glassfish, JBoss etc.

→ DATABASE CONNECTION

JDBC component connects to a data source defined by a URL format. URL contains 3 parts :-

(i) jdbc - It indicates that the jdbc protocol is to be used to read the URL

(ii) <subprotocol> it is a driver name

(iii) <subname> it is a name of a database.
(It has to replaced by a val.)

getConnection() : One of the ~~key db~~ 3 getConnection() is used to establish connection with the db. It requests access to db from DBMS. It is upto DBMS to grant/reject the access. A connection obj is returned if access is granted. else an SQLException is thrown.

Page No.
Date

(i) Sometime DBMS grants the access of db to anyone then only 1 parameter that is url is passed to getConnection().

(ii) Some DBMS allows access only to authorised users that need url, userid & password need to be supplied.

(iii) there may be occasions when a dbms needs some info other than userid & password to provide access to the db. These properties are stored in the 'Properties' object.

19/Sept/19

3 types of Statement Obj

① Statement - executes query immediately.
② PreparedStatement - executes a compiled query
③ CallableStatement - to execute StoredProcedures.

① for static queries and compiled everytime it executed.
② for parameterised queries, precompiled, (compile once; execute everytime)
③ eg.

```
PreparedStatement s = con.prepareStatement("update student set firstname = ? where rollno = ?");
s.setString(1, "Megha");
int numRows = s.executeUpdate();
```

↳ no. of rows affected

```
ResultSet rs = stmt.executeQuery("select * from Student");
while (rs.next())
{ ... }
```

IN OUT INOUT

Page No.
Date

③ CallableStatement s3 = con.prepareCall("{ call proc-name ( )}");
format func.

boolean t = s3.execute();
↳ whether the query is executable or not. (syntax)

⇒ Type of Queries for execution

* executeQuery() - used in ①&② Statement
it executes the query passed as an argument. returns ResultSet Obj's only.
Obj that contains rows, columns & metadata that represents data requested by the query

* executeUpdate() - " " " " "
inserts, updates, delete and DDL SQL statements. It changes the values in a row or removes a row respectively.

* execute() - used with all the three Statement Obj.
used to return T/F boolean values if the query is executable or not and a ResultSet can be retrieved or not.

⇒ Scrollable ResultSet

ResultSet → Type
→ Concurrency.

Types of ResultSet

1. TYPE_FORWARD_ONLY
2. TYPE_SCROLL_SENSITIVE
3. TYPE_SCROLL_INSENSITIVE

Methods used in ResultSet -
next(), first(), last(), previous()
absolute(3) → 3rd row
relative(-1) → current row - 1
By default only next() can be used.
getRow() - returns row no. of current row
Note Virtual cursor is initially 0.

3-1 = 2

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Concurrency

1. → CONCUR_UPDATABLE
2. → CONCUR_READONLY (by default)

2. ResultSet can be used for read only. getXXX();
1. ResultSet can be updated & sent it back to DBMS (update table).
we use updateXXX();

St s = con.createStatement (type of ResultSet);

⟹ DriverManager.setLoginTimeout(int seconds)
public static void method of DriverManager class.
It is used to establish the max. time that a DriverManager waits for a response from a DBMS before timing out.
We use this method so that rather than waiting for the delayed response from DBMS, the DriverManager can ~~send~~ cease to connect attempt to to the db.

⟹ public static int getLoginTimeout()
used to retrieve the max time that DriverManager is set to wait until it times out.

⟹ Why do we use PreparedStatement objs?

Page No.
Date

① The compilation of an SQL query is performed after one of the execution methods of Statement Object is call. Compiling a query is an overhead that is ~~exp~~ accepting if the query is called once but it is expensive if it is executed many times by the same program in a session. PreparedStatement Obj can be used to precompile & execute an SQL query.

② (?) is used as an placeholder for a value that is inserted into the query after the query is compiled. This value changes each time the query is executed.

③ PreparedStatement() of Connection Obj: returns Prepared Statement Obj if it is passed an query which is then precompiled.

④ setXXX() of PreparedStatement Obj are used to replace (?) with actual value passed to the method. XXX is a datatype of the value passed. Eg. setString() is used to replace ? with a string value. Parameters - parameterIndex & value.

(done at Runtime) ← PreCompilation done by DBMS is also called Late Binding.

When DBMS receives a request it tries to match the passed query to a pre compiled query. If the match is found the parameters passed to query using setXXX() are bound & query is executed.

If the match is not found the query is compiled & retained by DBMS for later use.

⇒ JDBC driver passes & parameters to DBMS:-

(i) Query (ii) An array of late Binding Values.

⇒ Callable Statement - used to call a stored procedure from within a java program. A stored procedure is a blk of code & is

identified by a unique name. The Syntax of stored procedure is specific to the DBMS. The CallableStatement Obj uses three types of parameters to call a stored procedure. These are :- IN, OUT and INOUT.

IN parameter contains the data to be passed to the stored procedure. And whose values are assigned using SetXXX().

OUT parameter contain the values returned by the stored procedures. These parameters are registered using registerOutParameter() & retrieved using getXXX().

INOUT parameter is a single parameter that is used to pass info to the stored procedures as well as retrieve info from a stored procedure. The same methods as above are used to make use of INOUT parameters.

Eg. MySQL

```
delimiter $$
create procedure proc-name(
    OUT num int)
    select max(rollno) from Student;
$$
```

Query - JAVA

```
String query = "{ call proc-name() }";
CallableStatement cs = con.prepareCall(query);
```

cs.registerOutParameter (1, TYPE-INT);
Out no. parameter no.

cs.execute();

int maxroll = cs.getInt(1);

05/Sept/19

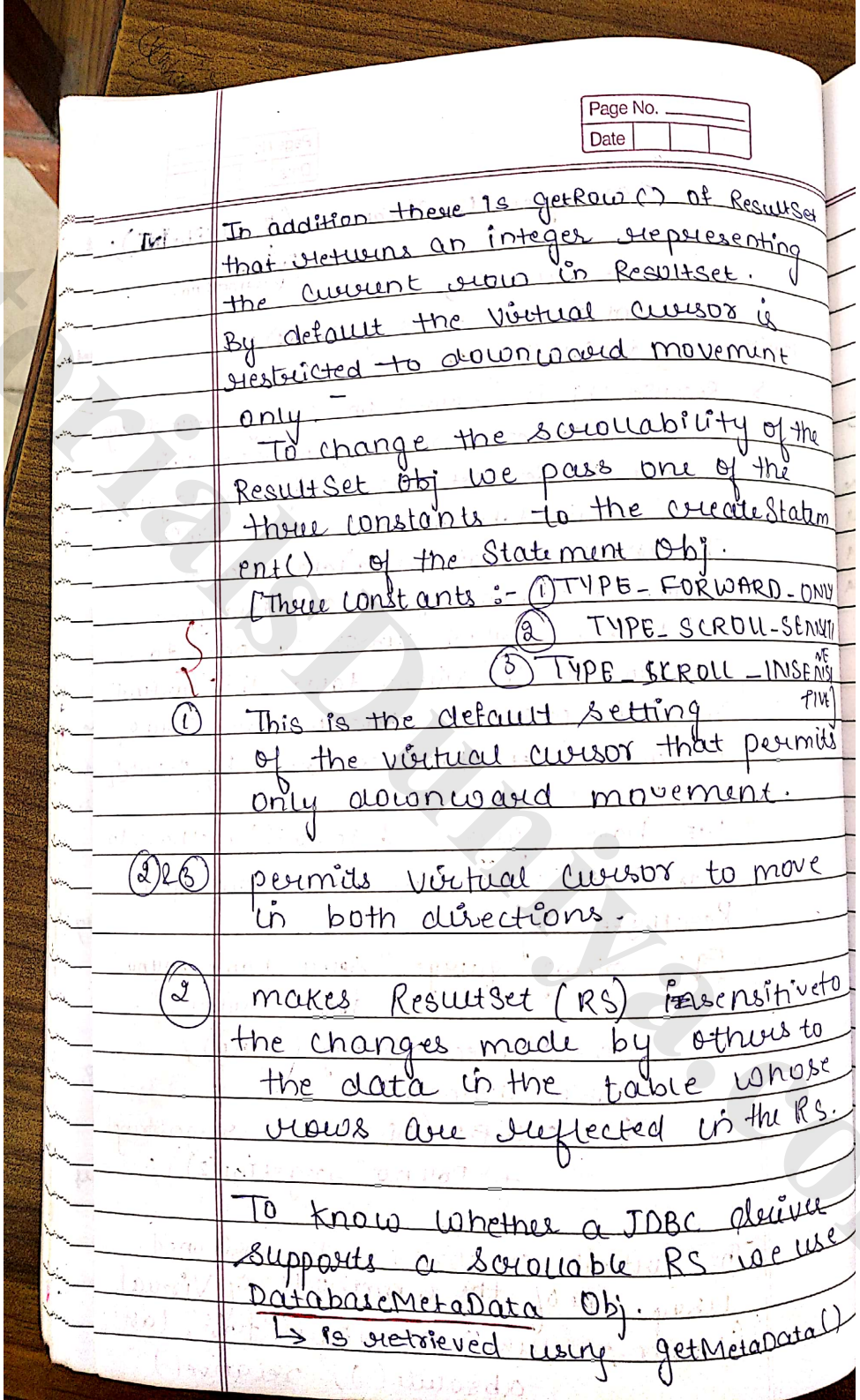
RESULTSET TYPE

⇒ ResultSet obj is returned on calling executeQuery() after it processes an SQL query passed to it. Data in the ResultSet obj is logically organised as a virtual table containing rows & col^n's. This obj also contains metadata such as column names, col^n size, & col^n datatype. It uses a virtual cursor to point to a row of the virtual table. This virtual cursor is positioned above the first row of data in ResultSet when returned by executeQuery().

We have getxxx() & setxxx() methods to retrieve or set values in a row of the ResultSet Obj.

Eg: String query = "select name, rollno from student;";
rs = s.executeQuery(query);
while(rs.next())
{ S.O.P.L("Name:" + rs.getString(1) + "Roll no:" + rs.getInt(2));

The virtual cursor can be positioned using one of the 6 methods of Virtual Cursor:- next(), previous() first() last() relative()



Page No.
Date

In addition there is getRow() of ResultSet that returns an integer representing the current row in ResultSet. By default the virtual cursor is restricted to downward movement only.

To change the scrollability of the ResultSet Obj we pass one of the three constants to the createStatement() of the Statement Obj.

[Three constants :- (1) TYPE_FORWARD_ONLY
(2) TYPE_SCROLL_SENSITIVE
(3) TYPE_SCROLL_INSENSITIVE]

(1) This is the default setting of the virtual cursor that permits only downward movement.

(2)&(3) permits virtual cursor to move in both directions.

(2) makes ResultSet (RS) sensitive to the changes made by others to the data in the table whose rows are reflected in the RS.

To know whether a JDBC driver supports a scrollable RS we use DatabaseMetaData Obj.
↳ is retrieved using getMetaData()

of connection obj. [Eg]

DataBaseMetaData meta = con.getMetaData();

We call supportsResultSetType() of this metaData obj. to test whether the RS type passed to the method is supported or not.

boolean sensitive = meta.supportsResultSetType (ResultSet.TYPE_SCROLL_SENSITIVE);

* setFetchSize(int) → Retrive two rows of RS at a time.

When the rows retrieved in the RS are in huge no.s then we can allow the driver to request a fixed size of the rows fetched from the db at a time. for this we call setFetchSize() of the RS to set the fetchSize.

⟹ Concurrency / Updatable RS

Rows in a RS can be updated just like updating rows in a db table. for this ResultSet.CONCUR_UPDATABLE const is passed to the createStatement().

To prevent any updation CONCUR_READ_ONLY const is passed. this is also the default setting of the RS concurrency.

ResultSet

Page No.
Date

Update ResultSet

(1) Modification (2) Insertion (3) Deletion

(1)
```
ResultSet rs = st.executeQuery("
    Select rollno,name from Student");
rs.absolute(2);
rs.updateString
    ("name", "Sam");
rs.updateRow();
```

In ResultSet →

In actual db →

RowNo.	Rollno.	Nam
1	101	Megh
→2	102	Nish SAM
3	103	Snm

↑(ResultSet)

```
// Printing the rows.
rs.absolute(0);
while (rs.next())
    S.O.P.L (rs.getInt(1) + rs.getString(2));
```

(2)
```
rs.moveToInsertRow(); // move virtual ptr to (last+1) row.
rs.updateString("name", "sam");
rs.updateInt("rollno", 120);
rs.updateNull(colno.);
rs.insertRow();
```

Set the col n value to null ←

(3)
```
rs.absolute(2);
rs.deleteRow();
```

(1) After executeQuery returns RS obj

Atomicity - A
Consistency - C
Isolation - I
Durability - D

Page No.
Date

updateXXX() is used to change the value of the col^n in the current row of the RS. XXX denotes the datatype of the col^n to be updated. It needs 2 parameters:-
(i) col^n no. or name to be updated
(ii) value that will replace the current col^n value.

updateNull() used to replace a col^n with a null value. It has only one parameter that is the col^n no. in the current row of RS. It does not accept col^n name.

(2) It uses the same technique as used in the updation of the RS.
First moveToInsertRow() of RSObj is called to reach to the insertRow then updateXXX() are called followed by one insertRow call.

(3) used to remove a row from the ResultSet.

Note- Any updation, insertion or deletion using updateRow, insertRow or deleteRow() affects the underlying db.

Page No.
Date

⇒ Transaction set of tasks. It gets completed successfully only if each of the task get completed. If one task fails then the entire task fails. In that case all previously completed tasks must be reversed i.e. rolled back. A db transaction consists of a set of SQL statements.
commit() of the Connection Obj completes a db transaction.

Note

The DBMS has an autoCommit feature that is bydefault set to true in that case we do not need to call commit().
To deactivate the autoCommit feature we call - setAutoCommit() of Connection Obj & pass false boolean value to it.

EXAMPLE -

```
try{
con.setAutoCommit(false);
Statement s = con.createStatement();
int s_num_of_rows = s.executeUpdate("update
                    Student set name="Pito"
                    where roll no=80);
s.setInt(1,101);
con.commit();
}
catch(SQLException e)
{
try{ if(con!=null)
     con.rollback();
e.printStackTrace();
```

Page No.
Date

```
catch(SQLException e) //if rollback() is not executed.
{   s.o.p(ex.getMessage());
}
}
```

Note To prevent all the task in a transaction to rollback & let only a specific no. of tasks to be rollback we use **Savepoints**. To create a savepoint we call **setSavepoint("")** of connection obj & pass a savepoint name. This method returns a Savepoint obj. After completion of all the statements post the Savepoint we call releaseSavepoint() method of Connection obj. passing the Savepoint name.

In the rollback call we pass the savepoint name.

EXAMPLE-

```
try{
    con.setAutoConnect(false);
Statement s1 = con.createStatement();
Savepoint sp = con.setSavepoint("sp1");
Statement s2 = con.createStatement();
int n = s2.executeUpdate(query2);
    con.commit();
    con.releaseSavepoint("sp1");
    s1.close(); s2.close(); con.close();
}
catch(SQLException e)
{       :
    con.rollback("sp");
}       :
```


Visit TUTORIALSDUNIYA.COM for Notes, Tutorials, Programs, Question Papers

Page No.
Date

ResultSet Holdability -

We can control whether the RS obj need to be closed or not after the call to commit(). This is done by passing one of the 2 constants to createStatement().

(i) HOLD_CURSORS_OVER_COMMIT
This keeps RS obj open.

(ii) CLOSE_CURSORS_AT_COMMIT
This will close RS obj on commit.

Batch

To combine SQL stmts into a transaction is to batch them together. This is done by calling addBatch() & passing the SQL query string of connection obj
After adding all the stmts we call executeBatch() of a stmt. obj to run the entire batch at the same time.
It returns the integer array that contains the no. of SQL stmt's executed successfully.
BatchUpdateException - If an error occurs during the execution of batch.
clearBatch() - Clear the entire batch created.

→ 3 types of Exception

Visit TUTORIALSDUNIYA.COM for Notes, Tutorials, Programs, Question Papers

* SQLException - syntax errors in sql query, connectivity issue, any coding errors, such as accessing a closed obj.
* SQLWarningEx - throws warning received by connection obj. from dbms.
* DataTruncationEx - Whenever data is lost due to the truncation of data value, DataTruncationException is thrown.

getMetaData() of connection obj is used to retrieve a Database MetaData interface obj which contains info about dbs, tables, col^ns, indexes etc. about the dbms.

⇒ DatabaseMetaData Methods

getUserName()
getDatabaseProductName()
getSchemas()
getPrimaryKeys()
getTables()

Page No.
Date

SQLException - Syntax error in sql query, connectivity issue, any coding errors, such as accessing a closed obj.

SQLWarning Ex - throws warning received by connection obj. from dbms.

DataTruncation - Whenever data is lost due to the truncation of data value, DataTruncationException is thrown.

getMetaData() of connection obj is used to retrieve a Database MetaData interface obj which contains info about dbs, tables, col^n's, indexes etc. about the dbms.

⇒ **DatabaseMetaData Methods**

getUserName()
getDatabaseProductName()
getSchemas()
getPrimaryKeys()
getTables()

UNIT-4 JSP → Server Side technology

* JSP → Java Server Pages. Used to display dynamic content. It uses Client Server → Request Response model
* MVC - Model View Controller Model
JSP is View in MVC design ↓ Design/Pattern

JSP is translated into servlets.

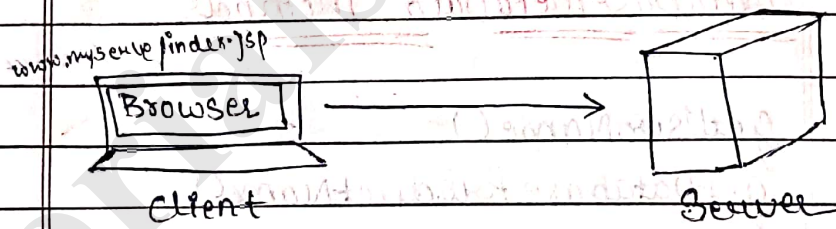
Page No. _____
Date

→ do separation of concerns

⇒ MVC divides the data into three parts:-
* Model - business logic (Process the data)
* View - GUI (display to Client)
* Controller - Intercepts the request
↳ It is first interface that receives the request. It decides what to do with data.

⇒ Java EE API (EE - Enterprise Edition)
* Model part is EJBs (Enterprise Java Beans)
↳ Process data ↳ NIC (Not in course)
* JSP is a View.
* Servelet is the controller.
↳ It handles Generic or HTTP requests.

⇒ HTTP



Request Message / Response Msg.

Request Msg
- Request Line → Protocol, Version, resource, method.
- Request Headers → accept: image/jpeg, accept-language/en
- Request Body → optional

Method - get, post, Delete, Trace, Head, Request

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

Response line (RL)
" Code (RC)
" Header
" Body → Actual Response

RC - Whether Request is accepted or rejected
Eg. 404, 200 (Accepted)

⇒ Difference b/w Servelet & JSP

* Servelet extends functionality of web server. In small process it also processes the request. It returns response & fetch request.

⇒ Life cycle of JSP page.

* fetch request by Servelet
* Translation - jsp page to Servlet.
* Initialization
* Execution
* Clean-up

Request for jsp
↓
Translation
↓
Initialization
↓
Execution ← Subsequent Next Request
↓ Response to Client
Cleanup

⇒ Packages

* java.servelet.jsp.* (for generic servlet)
* java.servlet.http.HttpServlet.* (for http)
* javax.servlet.jsp.* (for jsp page)

Page No.
Date

Note: Servlet Container creates servlet class. Automatically imported by " "

⟹ Structure of Servlet Class

Methods in jsp
* init() → Initialization
* service() → execution
* distroy() → clean up

Methods in Servlet
* _jspInit() ≃ init() of jsp.
* _jspService()
* _jspDestroy()

⟹ JSP Page Anatomy

says jsp to do something → attribute

```
<%@page import = "java.util.*" %>
<HTML>
<HEAD> <TITLE> ... </TITLE> </HEAD>
<BODY> <H2> Hello World ! </H2> </BODY>
</HTML>
```

Directive
* Page
* include
* taglib

Directive → It is a processing instruction to the servlet container.
for multiple imports separate them with a ;
We can have multiple directive & imports in a same file.

⟹ JSP Page — It has 2 parts —

* Template Text :- Std markup lang tags covers static part
* JSP Elements - contains java codes & covers dynamic part.
  ↳ Starts with <%.

Page No. _____
Date

⇒ Scripting elements :-

* Scriplets — <% ... %> (Java code) — int x = ... Scripting code
* Expressions — to output a literal value of expression value — <%= ... %>
* Declarations — used for methods or var declaration — <%! int sum() { } %>

⇒ CODE

```
<%@ page language = "java" %>
<HTML> <HEAD> </HEAD>
<BODY>
<%! int sum (int x, int y)
        { return x+y; } 
%>
Hello, this page shows use of JSS
Scripting elements
<BR/>
<% out.println("<BR/> out is predefined
        Object of JSP Writer ");
%>

Sum of 5 & 6 is <%= sum(5,6) %>
or
<%= "Sum is :" + sum(2,3) %>
```

20/Oct/2019

Page No.
Date

⇒ Servlet Container - a web server.

WAR - Web Archieve file (appl^n)

Web appl^n are archieved into a war file. (zipped or)

↳ It contains libraries & class files

JAR - Java Archive

Life cycle of Servlet Container

(1) load the class & instantiation

loading is done by Servlet container.

(2) Initialization

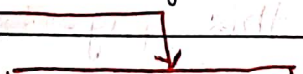
(3) Execution

(4) Clean-Up

loading + Initialization is Deployment.

⇒ Deployment Descriptor is web.xml which is (Configuration file)
It contains initial parameters, url mappings & other configurations

⇒ Context path



google.com / n1 / f23 / in.jsp ? —— & ——

host name. Appl^n or Context

⇒ JSP Page.

(1) Directive - JSP element which instructs the JSP container to do something. There are 3 directives - page, include & taglib.

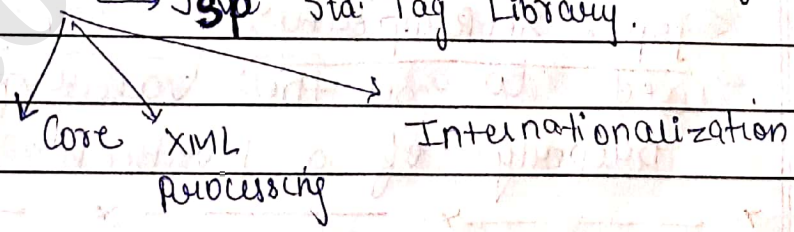
Page No. _____
Date

(i) Page directive - <%@ page ... %>
It specifies the info about the page itself that remains the same between the requests. It defines page dependent attr. such as session tracking, error page, buffering requirements etc. for eg -
<%@ page language = "java" extends = "myPaintClass.java" import = "java.sql.*" session = " " %>
The page directive includes one or more attribute-value pairs.

(ii) Include directive - used to include a file during the translation phase. for eg -
<%@ include file = "Ca.java" %> [only have one attr]

(iii) taglib directive - used to include a tag library containing custom tags which are used in the jsp page. for eg.
<%@ taglib prefix = "c" uri = "http://www.sun.com/jsp/jstl/core" %> core
<c:out value = "${1+2+3}" >

* JSTL - It is a std tag library.
↳ Jsp Std. Tag Library.



It is provided by Sun.

WS → Expression lang :

Page No.
Date

→ Scripting el - are used to embed pieces of code within a jsp page.

→ Action el - performs specific actions requested by a browser at runtime.
There are 2 types of action els -
(i) Std. action elements
(ii) Custom action elements and JSTL

(i) These elements perform common actions such as including the response of another JSP page within the current page, instantiating a javabean, forwarding a request to another jsp page etc.
It starts with <jsp:
<jsp:useBean>
It makes a javabean a/v in a java page.
<jsp:getProperty>
It is used to retrieve the value of a property of a javaBean instance.
<jsp:setProperty>
used to set the value of a property of a JavaBean instance.

× × × ×

* JavaBean - it is a java class that represents an entity. It has some attrs & properties, Accessor & mutator methods
It follows some coding conventions.

Page No.
Date

for eg: for each coln in Student table of a database there will a class having properties with the same name as coln & for each property we will have 2 methods - setXX() & getXX()

Student → rollno & name.

setrollno()
getrollno()

setname()
getname()

<jsp: include>

It includes another servlet or jsp page response within the current page during request processing phase.

eg - a1.jsp

```
<%@ include file="hello.html"%>
<HTML> <BODY>
Abcd
</BODY> </HTML>
```

hello.html

```
<HTML> <BODY>
Hello World!
</BODY> </HTML>
```

→ it merge code during ~~translation~~.

or

```
<jsp: include ~~file~~ page="hello.html" >
```

<jsp: forward> It forwards the request processing to another servlet or jsp page.

<jsp: param> it adds a parameter value to the request handed over to another jsp page or servlet using <jsp: include> or <jsp: forward> action elements.

<jsp:plugin> → generates the html that contains appropriate browser dependent elements (OBJECT or EMBED) which are needed to execute an applet with java plugin s/w.

(ii) Custom Action Elements are defined by the jsp specification to extend the jsp language either as java classes or text files with jsp elements.
JSTL is one such extension defined by Sun and bundled with jsp container. It has features needed by most of the jsp appl^n such as conditional processing, db access etc.
Other than this programmers can also define additional custom tags.

Q. Difference b/w <jsp:include> and include director (<%@include ...%>)

<jsp:include>	<%@ include ... %>
① it includes the resource at the request processing time	includes the resource at the translation time of a jsp page
② better for dynamic pages.	better for static pages.

Page No.
Date

(3) Syntax: <jsp:include page = "<relativeurl>"/>	Syntax: <%@ include file = "<relativeurl>" %>
(4) This action element corresponds to a method invocation within the servlet	It includes the original content of the resource in the generated servlet.

⇒ Expression Language (EL) Expression (4th JSP Element type)

* originally developed as a part of JSTL specification.
* It is a simple lang for accessing the user request data and data of the appln classes.
* It can be used directly in template text. or to assign values to action element attributes.
* It is not a fully fledged programming lang & is similar to javascript.

⇒ JavaBean - is a java class that complies with specific coding conventions. It is used as a containes of info that describes the entities of an appln. for eg - Student, Customer, Order etc.

⇒ Deployment Descriptor - All the web Appln resources along with a deployment descriptor as arranged in a well defined hierarchy & placed in a file called web Appln Archive (WAR). WAR file has

Page No.
Date

.war extension and it can be created using Java's jar command ([jar StudentManagementSystem] ↳ it will contains class file only.
↓
It will make war file)
or using a zip utility programme such as WinZip or using an IDE's tool.

Deployment descriptor is a file web.xml that contains the configuration info for an appln. It is an xml file within the WEB-INF directory of the appln. And it contains info about how all the resources fit together, security requirements, url mappings, initial parameters etc.
WEB-INF directory has 2 special sub-directories - lib and classes.
All the class files of the appln must be stored in these 2 directories.

lib directory contains all JAR files.
classes contains all the classes such as servlet or the custom tag library classes etc. which are not in JAR file.
All the directory & file name are case sensitive.

31/10/19.

⟹ Difference between Servlet & JSP

Date

* Servlets are server side JavaEE technology that provide extension to the server.
JSP is a server side JavaEE technology used as a scripting language.
* The lifecycle of a Servlet includes loading & instantiation of the Servlet class, initialization, request processing (execution) and clean-up.
The lifecycle methods of Servlet are init(), service() and destroy.
The lifecycle of a JSP page includes a translation phase, loading, instantiation of corresponding Servlet class, initialization, execution and clean-up.
The lifecycle methods are -jspInit(), -jspServiu() and -jspDestroy().
* A Servlet prousses a request using the same instance during its lifecycle. This instance get destroyed when the server is shutdown.
JSP requires one additional step of translation and compilation. This makes JSP slower than Servlet. However, it does not need to be recompiled across the request unless this jsp page is modified or server is shutdown.
* Servlets are more processing intensive & have less html text. In a Servlet html content is embedded within the java code. So, programmers find it suitable to write Servlets.
JSP contains more of html content than processing logic. Here, java code is embedded within html text. So, it is more suitable for web designers to write jsp pages than to write Servlets.

Page No.
Date

Bean – a bean is declared and instantiated using JSP useBean Std action ele * The JSP getProperty element reads the current value of a Bean property and inserts it into the response of http msg.

* An action ele can be used to set an html ele attr but it cannot set the value of another action ele

Eg – <img src='<jsp:getProperty name="ob" property="filename" width="100">' also wrong>

<jsp:setProperty name="ob" property="value" = "<jsp: getProperty . . . >" > // not allowed

* To set the jsp action attr to a value produced by another action, we must use <jsp:attribute> action ele Eg – class Cartoon { String filename; }

<jsp:setProperty name="c1" property="fileName" value>
<jsp:attribute name="value" trim="true">
<jsp:getProperty name="c2" property="filename" />
</jsp:attribute>
</jsp:setProperty>

⇒ <jsp:attribute>

* <jsp:getProperty> is nested with <jsp:attribute> action. <jsp:attr> can only be used in the body of another jsp ele.

* It cannot be used for template data elts.

Jsp page — template text → HTML, XML
JSP ele

Page No.
Date

* It evaluates its body. It then sets the attrs specified by its name attr of its parent action element to the o/p produced by the nested jsp ele within its body.
* This name attr of <jsp:attribute> action is mandatory the trim attr. is optional & true by default. if it is true then all the leading & trailing white spaces in the body is removed. Otherwise the result is used as is.
* We can use a jsp EL (Expression lang) to read a property value. Eg- <img src="${c1.filename}">

⇒ Expression lang (EL)-- It interprets a name of Bean variable followed by a dot & a property name as a request to get the value of the property from the Bean.

⇒ Alternative for setProperty
JSTL action used along with EL.
<c:set target="${c1}" property="filename" value="mickey.gif">

⇒ Automatic type Conversion

Simple Java datatype ⟵ String (by web, servlet container container, web server, JSP container)

A bean property can be of any java type but the programmer does not need to perform

Page No.
Date

type conversion when dealing with bean obj's on setting / getting values of their properties. Simple types are automatically converted into text values (String type) by the web container. But complex types such as user defined obj's need to be converted by the programmer using so called property editor for handling the conversions.

31/10/19

CHAPTER-7 Custom Tag Libraries & JSTL

* JSP std actions defined in JSTL are HTML like elements for commonly needed fn's in a JSP page such as creating beans, accessing bean properties, other JSP pages. But we can do a lot more for this we can develop new actions as Java classes or tag files (a special kind of jsp file) to extend the set of action eles. These actions are called custom actions. A custom action can perform anything. It has access to all the info about a request, it can add content to a response, can set response headers, and can use any Java API.

⟹ TLD (Tag library Descriptor)

The name of a tag or class file that implements a custom action & all

Page No.
Date

the info needed by the container to invoke it are specified in a file called TLD.

* A custom tag library is simply a collection of TLD and all classes / tagfiles for ~~Classes / tags~~ a related set of custom actions. A custom tag library containing these file is usually pkged in a jar file. to make it easy to install.

- for each TLD the container gets the default uri of the custom tag library from the TLD itself. The TLD creates a mapping of uri to the TLD that contains it. The uri attr of taglib directory matches this default uri.

- JSTL library is implemented in the same way as appln specific custom tag library. The only difference is that the functionality and syntax of jstl libraries are defined by a formal specification. There are 5 different JSTL tag libraries which divide all the actions into different categories.

URI for JSTL

Library	prefix	uri
Core	c	http://java.sun.com/jsp/jstl/core
XML processing	x	" " " " " /xml
I18N formatting	fmt	" " " " /fmt
Database Access	sql	" " " /sql
Functions	fn	" " " /functions

Page No.
Date

The first 4 categories are custom actions and the functions library contains a set of actions that can be envoked in a EL expression.

Pg.77 Eg- = "${fn:trim(varName)}"
↳ prefix

Eg- = "${fn:toUpperCase(String)}"

CH-8 Processing I/O & O/p.

Inorder to read the request parameter value, the web container creates an obj of javax.servlet.HttpServletRequest class that has properties & methods to access the request parameter. The user i/p data sent from a client is stored in this obj. But how the data is sent through http request depends on the request method. (get or post). for a get request parameters are sent as query string appended to the url while for a post request they are sent in the request body. Each parameter is represented by a name/value pair. This name comes from the name attr of the form ele and value is entered by the user or specified in the value attr.

Pg.84 Implicit param - Pg.

param is → EL var that represents a collection (java.util.map)

1) last para

2) <c:out> - Pg 85

<c:out value="" default="">

2.1) entity code

" → "
< → <
> → >
& → &
>= → ≥
nbsp →

2.2)

paramValues :-

checkbox { pizza, ... } food

<c:forEach items = "${paramValues.food}" var="curr">
<c:out value = "${curr}" />
</c:forEach>

Page No.
Date

7/11/19

⇒ Tag libraries — Std actions, Custom actions

(1) <x:hello msg = "Chavvu"> (mandatory)
Welcome to my page
</x:hello>

o/p: Hello Chavvu
Welcome to my page

⇒ Tag Extension Mechanism - import javax.servlet.jsp.tagext.*;
facility to use custom tages

Eg:-

Public class HelloTag extends SimpleTagSupport
{ Private String msg = "";
— it has only setter method —
Public void doTag() throws IOException
{
JspWriter out = getJspContext().getOut()
in class SimpleTagSupport — it returns obj of type PageContext;
StringWriter sw = new StringWriter()
It invokes the body of tage into sw.
getJspBody().invoke(sw);
out.println("Hello" + msg);

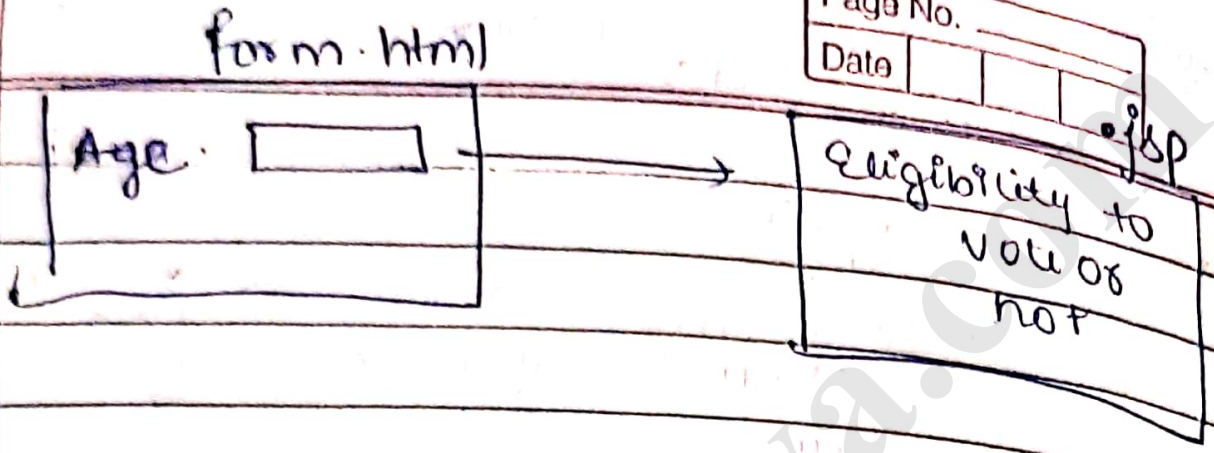
method of SimpleTag support class

⇒ TLD (Tag Library Descriptor) - TLD → WEBINF → web.xml
mypkg.HelloTag → mysrcpkg. (tlds → custom.tld)

→ <?xml
<taglib
<tlb version
<jsp version
<tag> <name> hello </name>
<tag-class> mypkg.HelloTag </tag-class>
<attribute> <name> message </name>
<required> true </required>
<type> String </type>
</attribute>
</tag>

Page No.
Date

(4)



port-3306 Download MySQL from community server and mysql java connector
↳ zip file extract in mySQL folder.

JSP

Page No.
Date

Q.1 Write JSP for use of directives.
Page directive – extends
include – file (html page o/p)
taglib – JSTL core library.
File → NewProj (Java Web) → Proj name

(2) Use of action elements.
<jsp: useBean> getProperty setProperty.

(3) userForm.html
Name :
Gender : ○ Male ○ Female
favourite food ☐ Pizza ☐ Pasta ☐ Chinese
Submit Data
userDetails.jsp?uname=sagar&gender=&food=
userDetails.jsp
welcome
Two objs
HttpServletRequest
HttpServletResponse
Collection
Map obj
(food["Pizza"])
① request in JSP ↳ Implicit obj
② param variable (Implicit 6)
③ Using jsp:useBean
<jsp:useBean class="" id="" scope="">
<jsp:setProperty name="" property="" value="">
<jsp:getProperty name="" property="">

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well