# WT QUESTION BANK(MID)

## UNIT-1

1. Write a java script to find factorial of a given number.
2. Briefly explain about JavaScript.
3. Explain about Events in JavaScript.
4. Explain Built in objects in JavaScript.
5. Explain different operators in JavaScript with suitable examples.
6. Explain conditional control statements in JavaScript. Give example.
7. Define function? Explain user defined functions. With suitable example.
8. Write a JavaScript code to validate a user by considering username as "ABC" and password as "XYZ". Assume username and password is getting from the form element.

## UNIT-II

1. Difference between SAX and DOM.
2. Explain in brief about XSLT.
3. Briefly explain the purpose of XML processor.
4. Explain in brief about DOM.
5. Briefly explain about XML with an example?
6. Define DTD? Explain types of DTD's with examples.
7. What is XML Schema? Explain with an example.
8. Differences between XML and HTML.

## UNIT –III

1. Write a program to print "Hello world!" using servlet.
2. What are the life-cycle methods for a servlet?
3. Define Servlet? Explain briefly.
4. Explain about lifecycle of servlet.

```
<html>
<script>
var n,i;fact=1;
n=parseInt(Window.prompt("enter a value"));
for(i=1;i<=n;i++).
{
fact*=i;
}
document.writeln("The factorial of n is:",+fact);
</script>
</html>
```

10/11/2024

# Introduction:

Java script was developed by Brendan eich in 1995, the language called Live Script and was later renamed javascript.

* Internet Explorer (IE) is the first web browser (VBB script before Java script only in IE)

After VB script .Net language (Not fully functional)

-> Script languages are used to design web pages.

Two types:
Client side (Java Script)
Server side

Javascript is a interpretted language

* Java Script can be run on any operating system and almost all web browsers.
* It is a client side scripting programming language used to create dynamic web pages content.
* Java Script is a lightweight (Interpreted bytecode ->), object based programming (Pre defined object) language.
* It is mainly used for 1) web development
  2) Web Application development
  3) Mobile Application development
  4) Smart Watch Applications
  5) IOT Applications and Microcontrollers

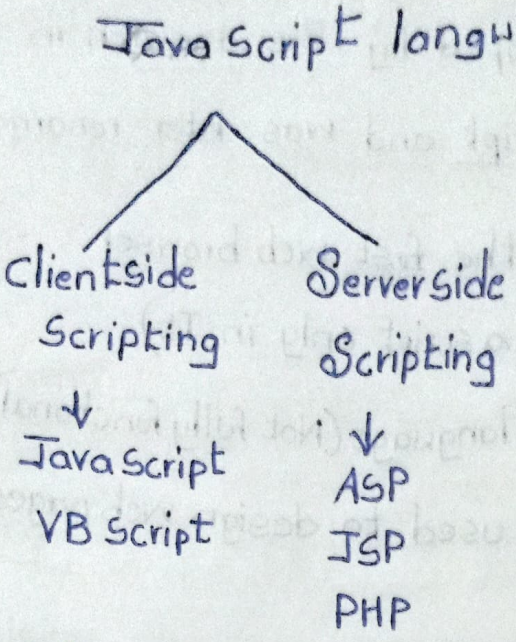
2

## Features

## Scripting Language:

There are specialized programming languages which are used to enhance functionality and appearance of web

Pages. There are two types:




=> Clientside scripting language used for validations at client side.

=> Serverside scripting languages used for database validation at Serverside.

Features:

- High level
- Dynamic
- Dynamically Typed
- Loosely Typed
- Interpreted
- Multi paradigm

## Event:

To change the state of an object is known as event there are various events which represents that some activity is performed by the user or by the browser. When java script code is included in html is react over these events and allow the execution this Process of reacting over the events is called Event Handling.

## Types Of Events:

1. OnClick Event:

It is used for responding for Mouse Click action When the user press a button it can call, any function through onclickEvent.

ex:

```
<input type = "Button" value = "ok" onclick = "func()">
```

→ Example:

```
<html>
<Script>
function fun()
{
alert ("Welcome to NRI");
}
</Script>
<body>
<input type = "Button" value = "ok" OnClick = "fun">
</body>
</html>
```

2. Onsubmit event:

The onsubmit event occurs when a form is submitted. It specifies an event handler function i.e., invoked when the user submit a form by clicking on a submit button in the form.

eg:

```
<html>
<script>
function fun1()
{
alert (" nri , name is submitted ");
}
</script>
<body> <input type = "text" name :"nriit">
<input type = "Button" value = "submit" onsubmit= "fun1()">
</body>
</html>
```

3. Onload event:

The onload event occurs when an object has been loaded. on load is most often used with the body element to execute a script once a web page is completely loaded all content.

The onload event can be used to check the visitors browsers type and load the proper version of the webpage based on the information.

example:

```
<html>
<script>
function fun( )
{
alert ("welcome to NRI");
}
</script>
<body onload = "fun ()">
</body>
</html>
```

# Objects in JavaScript:

Basic objects are 1) document object
2) Window object

## Document object:

* To display some information on the screen

Syntax: document.write("This is JS");
document.writeln("NRI is best");

In this, document is object name and write() or writeln() are methods.

4

</html>

-> Window objects

These are 3 types

1. Alert Box
2. Prompt Box
3. Confirm Box

Alert Box:

It is used to display message to user.

Syntax: Window.alert ("string msg");

Prompt Box:

Prompt Box is used to accepting data for variable

Syntax: window.prompt ("string msg", default value) (anything)

iii, Confirm Box :

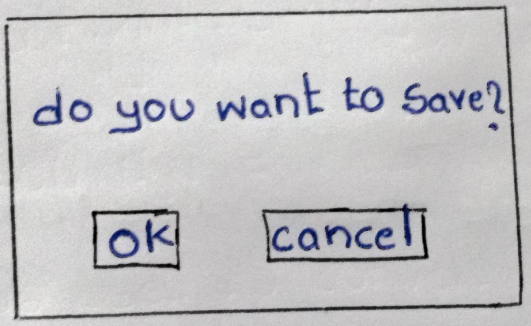
Window.Confirm( ):

It can be used to display a model window th

allows the user to confirm an action or provide feedback.

Syntax: window.confirm("message");

Ex: window.confirm("do you want to save?");




Math:

Syntax:

math.method (Numerical value);

1. min():

It displays the minimum value of two numeric values.

Ex: document.writeln(math.min(10,5)); // 5

2. max():

It displays the maximum value of two numeric values.

Ex: document.writeln(math.max(10,5)); //10

3. abs():

It returns the absolute value of a number.

Ex: document.writeln(math.abs(-43)); // 43

# String Object:

String refers to series of characters enclosed under double quotes(" ")

i, tolower(): tolowercase():

It is used to convert the given string into lower case letters.

Ex: var a="NRI";
document.write(String.tolowercase("NRI"));

ii, touppercase():

It is used to convert the given string into upper case letters.

Ex: Var a="nri";
document.Write(String.touppercase("nri"));

iii, Concat():

It simply combine or concates (concatenates) two strings.

Ex: var a="sumaja";
vab b="jangam";
document.write(a.concat(b));

v. getHours(): getMinutes(): g

5/2/23

# Array Objects:

Array is a collection of items or elements. In arrays are created using a special keyword "new".

Syntax: var arrayname = new Array();

var array1 = new Array(10);

## i, Push():

To insert the elements into an array/It is used to insert data into an array. once the data is pushed, array size gets increased.

Syntax:
```
var array1 = new Array(10);
array1.push();
```

array1.push(9); (9) → element; push → method; array1 → name of the array

## ii, POP():

It is used to remove elements from an array

Syntax: array1.pop(9);

## iii, sort():

It is used to arrange the elements in ascending order.

Syntax: array1.sort();

## Boolean Object :

It is an object wrapper for a boolean value.

The value passed as the argument to the constructor will be converted to a boolean value.

→ The value will be false if the provided argument is zero, '0'.

→ The value will be True if the provided argument is one '1'.

**Syntax :** var a = new. Boolean (Booleanvalue);

var a = new Boolean (true);

→ It accepts parameters true, false, 0, 1 (or) empty string.

If it is emptystring, NaN it treats as a false value.

The result is : true

24/11/24

## Conditional Statements: Conditional statements are

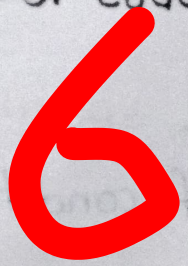
- Control behaviour in JS and determine whether or not piece of code can run

-> if statement
-> if-else statement
-> elseif statement

**if** : If a condition is true it is used to specify execution for a block of code.

Syntax:

```
if (condition)
{
Block of statements
}
```



Example:

```
<html>
<head>
<title> Even_Number </title>
<body>
<Script language = "Java Script" type = "text/javascript">
var a = parseInt(Window.prompt("enter a value:"));
if (a%2==0)
{
window.prompt("It is a even number");
document.Writeln("its a even number");
}
else
{
document.writeln("it is a odd number");
}
</script>
</body>
</head>
</html>
```

alert
alert - displays only content

**ifelse:** If the same condition is false it specifies the execution for a block of code.

**Syntax:**

```
if (condition)
{
Block of stmts
}
else
{
stmt
}
```

**elseif:** This specifies a new test if the first condition is false.

**Syntax:**

```
if (condition)
{
stmts
}
elseif (condition)
```

```
{
stmt
}
else
{
stmt
}
```

elseif:

Example:

```
<html>
<head>
<title> elseif example </title>
<body>
<Script language = "Javascript" type = "text /Javascript">
var a, b, c;
a = parseInt (Window. prompt ("enter a value"));
b = parseInt (Window. prompt ("enter b value"));
c = parseInt (Window. prompt ("enter c value"));
if (a>b && a>c)
{
alert ("a is big");
}
else if (b>c && b>a)
{
alert ("b is big");
}
else
{
alert ("c is big");
}
</script>
</body>
</head>
</html>
```

Syntax:

</html>

# Functions in JavaScript 7

Block of code used to perform specific task.

Func Funcname (Parameters list)
{
Block of code;
return value;
}

Syntax: → It is a block of code designed to perform a Particular task

→ A Java script function is executed when something invokes it.

→ Using functions, avoids repetition of the code as we need to write the code only once and then the code can be called anywhere using functionname.

functions into two types:

a. Built-in functions

## User defined functions:

JavaScript allows us to define our own functions as Per our requirement.

→ Function is defined by using the function keyword followed by the function name and paranthesis.

→ Function can be parametrised and non-parametrised.

## Calling Function:

After creating a function, we can call it anywhere in our script.

Syntax:

functionname (value1, value2, ...);

ex:

sum(5,6);

Return Statement:

A function may or maynot have a return statement because not all functions return an output.

→ A return statement is used to specify result or value ie.. returned from a function.

Syntax:

return value ;

ex:-

return x;

Program:

```
<html>
<head>
<title> Functionexample </title>
<script>
 function myfunction(a,b)
 {
  return (a+b);
 }
 myfunction(5,6);
</script>
</body>
</head>
</html>
```

<html><script
function validateUser() {
2) var username = document.getElementById('username').value;
var password = document.getElementById('password').value;
if (username == 'ABC' && Password = 'xyz')
{
alert ('login successful!');
}
else
{
alert ('Invalid username or password');
}
} </script>
</html>

This assumes you have a form with elements having the ids 'username' and 'password'. You can call this 'validate user' function when the form is submitted or when a login button is clicked.

5) <htm
<heac
<title
<bod
<Scri
docu
docu
docu
fo
do

</tr></table></form></body></html>

Differences between SAX and DOM

| SAX (simple API for xml) | DOM (Document object model) |
|---|---|
| 1. The SAX approach to Processing is called event Processing. | 1. The DOM approach to Processing for build the DOM file. |
| 2. This scans the xml docum -ent from beginning to end. | 2. This access the random parts of the xml document. |
| 3. The entire document accessed only once by the application | 3. If any part of the document must be accessed more than once by the application. |
| 4. It is faster than the DOM | 4. It is slower than the SAX. |
| 5. The SAX structure doesn't store entire document | 5. The DOM structure is stored entirely in memory for large document. |
| 6. It performs the syntactic analysis of the document. | 6. DOM requires SAX parser as a front end for analysis |

# XSLT- Extensible Stylesheet Language Transformation

Generally, the XML Document is Used to Describe the Data. We can Display the Data in XML Document Within the Browser By Using XSL(EXtensible Stylesheet Language). It Describes how the XML Document Should be Displayed.

⇒ To Display the Data in XML Document, XSL Creates a template.

→ XSL Basically, Transforms One datastructure to another. i.e., XML to HTML

## XSL Elements: xsl has 4 elements

1. <xsl:stylesheet>

It defines that this document is stylesheet document.

2

2. <xsl: Template match="/">

It defines a template that match with attributes associated with the root element of the XML Document.

3. <xsl: Value of>

Xsl value of can be used to extract the value of an xml element and add it to the output string stream.

4. <xsl: for-each>

Can be used to extract the value of an xml element repeatedly.

Example: Student.xml

```
<?xml version="1.0">
<xsl:stylesheet type="link/xsl" href="student.xsl">
<students>
<student>
<name> abc </name>
<rollno> 123 </rollno>
<branch> CSE </branch>
</student>
< student>
<name> xyz </name>
```

<rollno> 456 </rollno>
<branch > ECE </branch>
</student>
</students>

Student.xsl

```
<? xml version = "1.0">
<xsl: stylesheet xmlns = "http://603.org/xsl stylesheet">
-<xsl: template match = "/">
<html>
<body>
<h2> Student detail </h2>
<table boarder = "1">
<th > sname </th>
<th> Roll no </th>
<th> Branch </th>
<tr>
<xsl: for-each select = "students/student">
<td>
<xsl: value of select = "name"/>
</td>
<td>
<xsl: value of select = "Rollno"/>
</td>
<td>
<xsl: value of select = "Branch"/>
```

</td>
</tr>
</xsl:for-each>
</table>
</body>
</html>

[redacted]

</xsl:template>
</xsl:stylesheet>

## XML Processors:

The purpose of xml processors are as follows

=> The processor must check the basic syntax for well structureness.

=> The processor must replace all references to entities in an xml document.

=> DTD's and xml schemas can specify that certain values in an xml document have detail it values, which must be copied into the xml document during processing

=> Generally we have two processors for xml document for processing

=> Those are -> SAX (Simple API for xml)
-> (Document object model) DOM

3

9/3/24 - online

## XML DOM :

DOM is a platform and language that allocate Programs of scripts to dynamically access and update the content, structure and style of a xml document.

⇒ DOM defines the objects and properties of xml elements and the methods to access them

⇒ DOM exposes the whole documents to applications.

⇒ The parser reads xml into memory and converts into an xml DOM object that can be access with javascript.

⇒ Microsoft's xml parser is built into internet explorer

```
var xmlDoc = new ActiveXObject("microsoft-XMLDOC
xmlDOC.load("file.xml");
```

→ The first line creates an empty microsoft xml doc object

→ The second line creates the parser to load an xml doc called "file.xml".

### xml DOM properties:

→ x.node name → The name of x

→ x.node value

→ x.parent node

→ x.child node

→ x.attributes

Where x is the node object

4

x. removeChild(node) - remove a child

Ex: Users.xml

```
<?xml version="1.0">
<users>
<user>
<userid>1</userid>
<password>a</password>
</user>
<user>
<userid>2</userid>
<password>b</password>
</user>
</users>
```

login.html

```
<html>
<head>
<title>login</title>
<Script Language = "java script">
function validate()
{
var xmldoc = new ActiveXObject("microsoft.xmlDOC");
xmldoc.load("users.xml");
var id = f1.uname.value;
var pass = f1.pwd.value;
```

```
var x = xmldoc.getElementsByTagName("userid");
var y = xmldoc.getElementsByTagName("Password");
for(i=0;i<x.length;i++)
{
if(x[i].childNodes[0].nodevalue==id)
{
if(y[i].childNode[0].nodevalue==pass)
{
alert("successfully logged--");
return;
}
else
{
alert("Invalid password:");
f1.pwd.focus();
return;
}
}
}
alert("Invalid username--");
f1.uname.focus();
}
</script>
</head>
<body>
<form name="f1">
<br/>
<table align="center">
<caption>login</caption>
<tr>
<td>username:</td>
<td><input type="text" name="uname">
</td></tr>
```

```
<tr>
<td> password : </td>
<td> Input type = "password" name = "pwd">
</td> </tr>
<tr>
<td> <input type = "button" name = "reset" value = "login"
                                onclick = "validate()"> </td>
<td> <input type = "button" name = "reset" value = "Reset" </td>
</tr> </table> </form> </body> </html>
```

28/2/24

XML- Extensible Markup language It convert one form of data to another form

→ XML stands for Extensible Markup language used to store and carry the data near document.

→ It is the Extension of HTML.

→ Generally, HTML is restricted to only discipline the data.

→ XML is a metalanguage that describes contents of the document. so in this tags can be called as self describing

→ So data tags.

→ General syntax, of XML document

<?XML version = "1.0">

Example:

<root element>
<child element>
<sub-child element>...</sub-child>
</child element>
</root>

→ The XML code can be written on a simple notepad and should be saved as filename.XML

5

Ex:

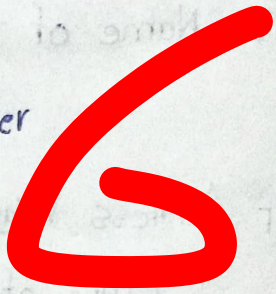
```
<document>
<student>
<name>Abc</name>
<rollno> 86 </rollno>
<age> 19 </age>
</student>
<student>
<name> def </name>
<rollno >99 </roll no>
<age > 20</age>
</student>
</document>
```

# XML DTD:

An XML DTD can be either specified inside the document, it can be kept in a seperate document and then linked seperately.

## Syntax:

```
<!DOCTYPE  element  DTDidentifier
    Declaration1;
    Declaration2;
    - - - - ---
    - - - - - -
>
```



→ The DTD starts with DOCTYPE delimitter.

→ An element tells the parser (library file/package, XML compiler) to parse the document type definition which may be the path to a file on the system or URL to a file on the internet.

→ If the DTD is pointing to external path it is called external DTD or external subset.

→ The square brackets ([ ]) enclosed an optional list of declaration called internal DTD or internal subset.

## Internal DTD:

→ Elements are declared within the XML file is called internal DTD.

→ To refer it as internal DTD, standalone attribute in the XML declaration must be said to "yes".

```
<?xml version = "1.0" standalone = "yes">
<!DOCTYPE address[
<!element name (#Pcdata)>
<!element D-no (#Pcdata)>
<!element city (#Pcdata)>

]>
<address>
<name> abc</name>
<D-no> 1-23</D-no>
<city> Vij </city>
</address>
```

External DTD:

→Declared outside the XML file.

→They are accessed by specifying the system attribute which may be either the legal DTD file or a valid URL.

→To refer it as an external DTD standalone attribute in the XML declaration must be set as no i.e.., declaration includes information from the external source.

```
<!DOCTYPE address
<!ELEMENT name(#PCDATA)>
<!ELEMENT D-no(#PCDATA)>
<!ELEMENT city (#PCDATA)>

> address.dtd
```

<?xml version = "1.0" standalone = "no" system "address.dt

<address>

<name>abc</name>

<D-no>1-23</D-no>

<City> vij </city>

</address>

## XML Schema:

→ An XML schema describes the structure of an XML.

→ XML documents can have a reference to a dtd or to an XML schema.

→ The XML schema language is also referred to as XML schema type definition.

→ The purpose of schema is to define the legal building blocks of an XML document, just like a dtd.

## An XML schema:

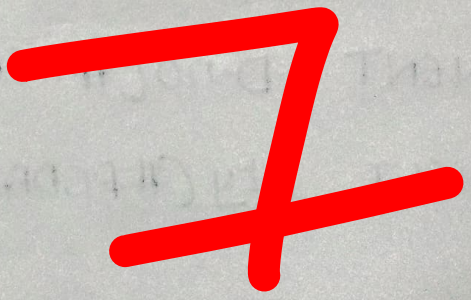
→ Define elements that can appear in a document.

→ Define attributes that can appear within the elements.

→ Define ~~switch~~ which elements are child elements and the sequence in which the child element can appear.

→ Defines the no of child elements and whether an element is empty or can include text

→ Define default values for attributes.



## Structure Of XML Schema:

```
<?xml version="1.0">
<xs:schema>
  ---------
  ---------
</xs:schema>
```

The schema element is the root element of the every XML schema, it can contain some attributes.

Ex:

```
<?xml version = "1.0">
<xs: schema xmlns = "http.//www.org//2001//xmlschema//">
<xs: element name = "employee">
<xs: complexType>
<xs: Sequence>
< xs: element name = "firstname" type = "xs:string">
<xs: element name = "lastname" type = "xs:string">
</xs: Sequence>
</xs: complextype>
</xs: Schema>
```

xml namespace

->XML declaration

Differences between XML and HTML

| XML | HTML |
| --- | --- |
| 1. XML stands for Extensible Markup Language. | 1. HTML stands for Hypertext Markup Language. |
| 2. User defined tags. | 2. Pre defined tags. |
| 3. User has control on tags. | 3. User has no control on tags. |
| 4. Root element is user defined and only one root element is allowed. | 4. Root element is HTML. |
| 5. We can generate new markup langu-ages using XML. | 5. No such possibility. |

| | |
|---|---|
| 6. Case sensitive | 6. Not a case sensitive |

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.servlet.*;
```

```
Public static void main (String args[])

{
class servletexample extends httpServlet

{
Public void doGet ( ServlethttpRequest req,
                    ServlethttpResponse res) throws ServletIOE-
                                                    xception
{
printWriter a = req.getprintWriter( );
a.println("Hello World");
}
}

}
```

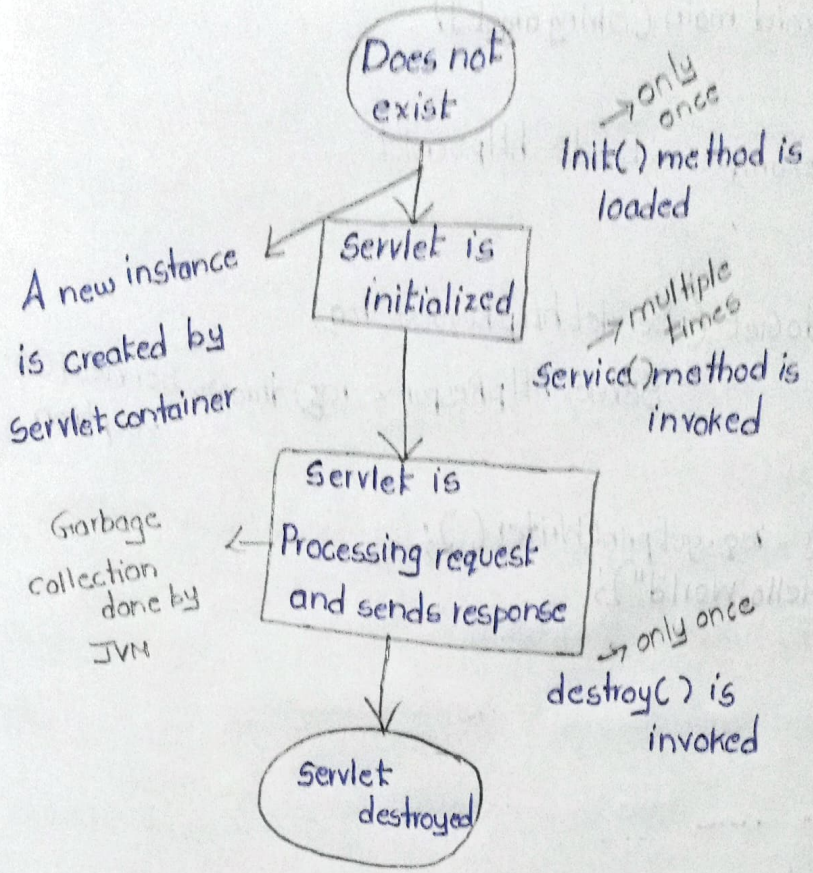
## Lifecycle of Servlet:

Lifecycle of the servlet describes how and when a servlet is loaded, initialised, able to handle request and unloaded (destroyed).

→ There are three methods in lifecycle of a servlet They are;

1. init()
2. service()
3. destroy()

2, 4




i, init()

This method initialize the servlet.

ii, service()

It process the request and provides the responses.

iii, destroy()

It is used to destroy the servlet.

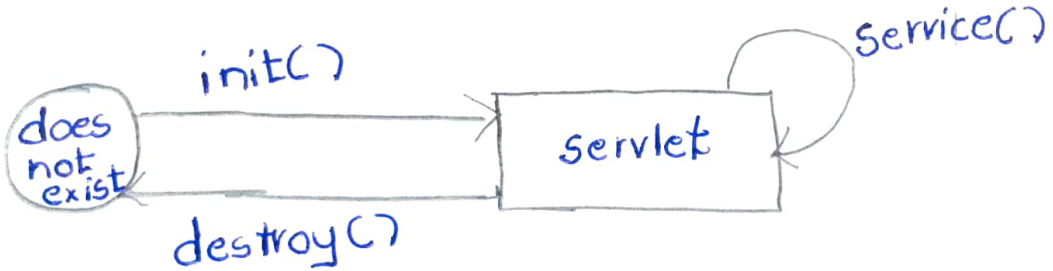
→ The servlet life cycle consists

1. The servlet class is loaded by the container during startup or the first time it is accessed.
2. The container calls the init() method this method is initialise the servlet and the init() method is called only once.

3. After initialization, the client request can be processored by the calling of service() method. The container can call the service method for every request.

4. Finally, if the servlet is not needed more, the container calls the destroy() method to stop the service and this method also called only once in the lifecycle of the servlet.

The simplified lifecycle of the servlet:

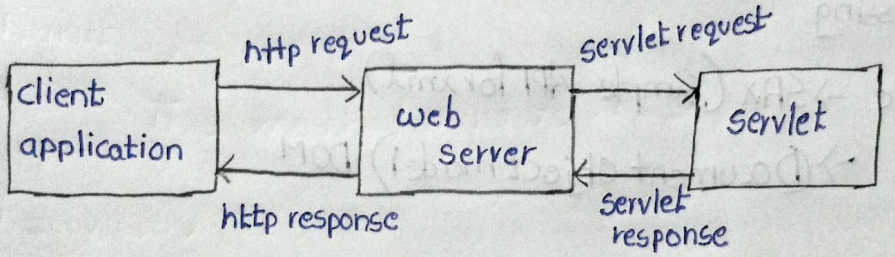


12/3/24 UNIT-3

## Servlet:

It is a java program that was created and executed by the web server. The servlet is used to execute the process at server side.

→ Servlet accepts a request from the client, performs some task and returns result to client. (webbrowser or a web application) The servlet runs on the server and will respond to a request from the client either in the form of a html pages.




→ To develop & run servlet we need following

1. Java Development kit (JDK) Installation
2. Java Servlet Development kit (JSDK)
3. A web server
4. A client application (Generally, A web browser)

## Basic Servlet Program:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.servlet.*;
```

3

```
Public static void main (String args[ ])
{
class servletexample extends httpServlet
{
Public void doGet ( Servlet http Request req,
                    Servlet http Response res) throws Servlet IOE-
                                                     xception
{
printWriter a = req.getprintWriter ( );
a.println ("Hello World");
}
}
}
```