MET's Institute of Engineering
Bhujbal Knowledge City, Adgaon, Nashik.
Department of Computer Engineering

"Java ServletS and XML"

Prepared By

Prof. Anand N. Gharu

(Assistant Professor)

Computer Dept.

CLASS : TE COMPUTER 2019

SUBJECT : WT (SEM-II)

UNIT : III

13 March 2023

SYLLABUS

Servlet: Servlet architecture overview, A “Hello World” servlet, Servlets generating dynamic content, Servlet life cycle, parameter data, sessions, cookies, URL rewriting, other Servlet capabilities, data storage, Servlets concurrency, databases (MySQL) and Java Servlets. **XML**: XML documents and vocabularies, XML declaration, XML Namespaces, DOM based XML processing, transforming XML documents, DTD: Schema, elements, attributes. **AJAX**: Introduction, Working of AJAX.

Syllabus

Servlet:

- Servlet architecture overview,
- Servlet life cycle,
- A “Hello World” servlet,
- Servlets generating dynamic content,
- parameter data,
- sessions, cookies, URL rewriting,
- other Servlet capabilities,
- data storage, Servlets concurrency, databases (MySQL) and Java Servlets.

XML:

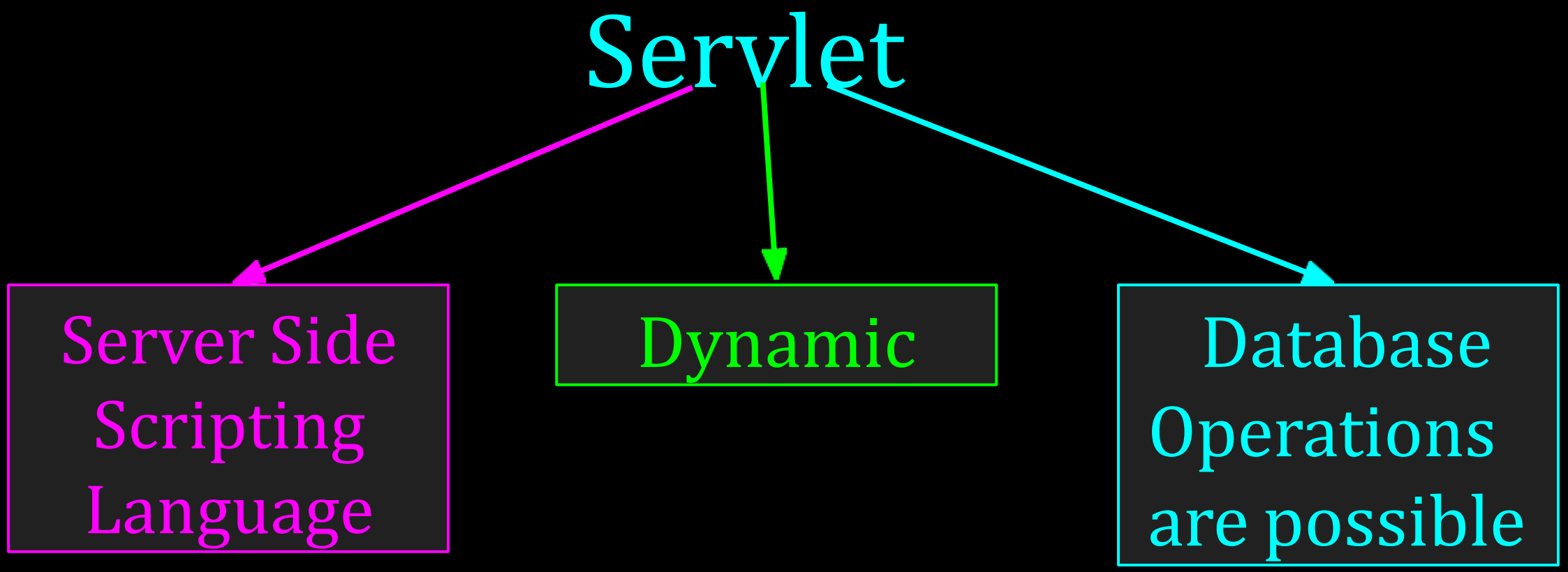
- XML documents and vocabularies,
- XML declaration,
- XML Namespaces,
- DOM based XML processing,
- transforming XML documents,
- DTD: Schema, elements, attributes.

AJAX:

- Introduction,
- Working of AJAX.

SERVLETs

Java Servlets

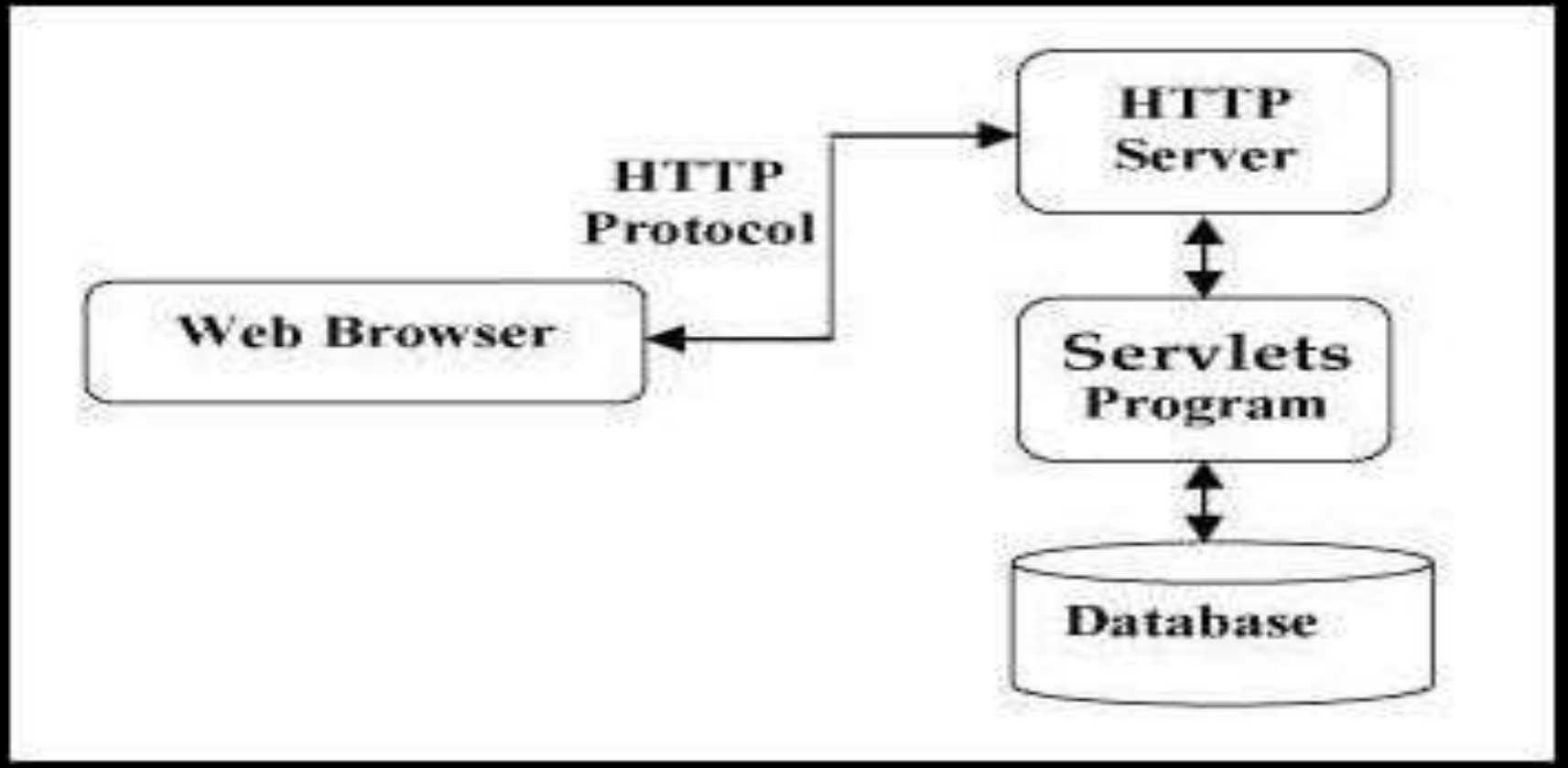


What are Servlets?

- Java Servlets are programs that run on a Web or Application server and act as a middle

  layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.
- Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.

Servlets Architecture

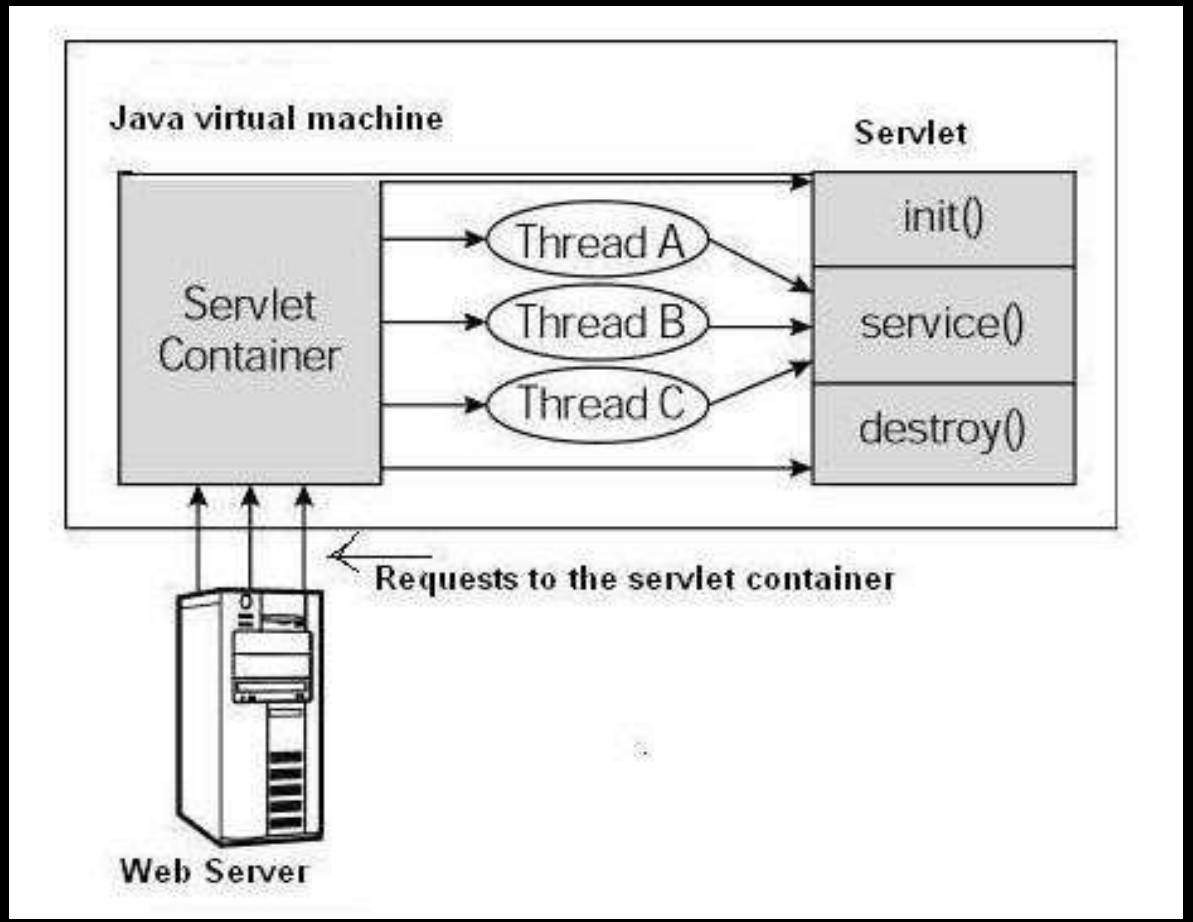


Servlet Life Cycle Diagram

First the HTTP requests coming to the server are delegated to the servlet container.

The servlet container loads the servlet before invoking the service() method.

Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



Servlet Life Cycle

- The servlet is initialized by calling the init() method.
- The servlet calls service() method to process a client's request.
- The servlet is terminated by calling the destroy() method.
- Finally, servlet is garbage collected by the garbage collector of the JVM

The init() Method

- The init method is called only once.
- It is called only when the servlet is created, and not called for any user requests afterwards.
- So, it is used for one-time initializations.
- When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate.
- The init() method simply creates or loads some data that will be used throughout the life of the servlet.
- The init method definition looks like this –

```
public void init() throws ServletException {
  // Initialization code... }
```

The service() Method

- The service() method is the main method to perform the actual task.
- The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the response back to the client.
- When server receives a request for a servlet, the service() method checks the HTTP request type (GET, POST) and calls doGet, doPost, methods as appropriate.

- Here is the signature of this method –

```
public void service(ServletRequest request, ServletResponse response)
 throws ServletException, IOException {}
```

The doGet() Method

- A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws ServletException, IOException {
  // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

The destroy() Method

- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close database connections, halt background threads, and perform other such cleanup activities.
- After the destroy() method is called, the servlet object is marked for garbage collection.

- The destroy method definition looks like this –

```
public void destroy() {
  // Finalization code...
}
```

Example 1-

To Print Hello World directly

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {

 public void init() throws ServletException { }

 public void doGet(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException
{
        response.setContentType("text/html");
       PrintWriter out =response.getWriter();
       out.println("<h1> Hello World </h1>");
   }
   public void destroy() { }
}
```

Reading Form Data using Servlet

- getParameter() – You call request.getParameter() method to get the value of a form parameter.
- getParameterValues() – Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- getParameterNames() – Call this method if you want a complete list of all parameters in the current request.

Example 2-

To read data from HTML file and print that .

- It requires 2 files:
  - 1. HTML (s1.html)File
  - 2. Servlet (s2.java) File

- First Run HTML file and after clicking on submit button it will run servel(.java) file.

Example 2-

To read data from HTML file and print that .

S1.html (html code)

```
<html>
<form method="post"
action="s2"> Enter Your Name
<input type=text name="t1">
<br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Example 2-

To read data from HTML file and print that .S2.java (Servlet Code)

```
import java.io.*;  import
javax.servlet.*;
public class s2 extends HttpServlet {
 public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
 IOException
 {
response.setContentType("text/html");
String a= request.getParameter("t1");
PrintWriter out= response.getWriter();
out.print("<br>Your Name is: "+a);
 }
}
```

Session Tracking (Management)

- **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques.
- Each time user requests to the server, server treats the request as the new request.
- So we need to maintain the state of an user to recognize to particular user.

- **Why use Session Tracking?**
- **To recognize the user** It is used to recognize the particular user.

Session Tracking Techniques

- Cookies
- Hidden Form Field
- URL Rewriting
- HttpSession

Session Tracking- Using Cookies

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- **How Cookie works**
  - In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, **we recognize the user as the old user.**

Session Tracking- Using Cookies

- **Types of Cookie**
  - Non-persistent cookie
  - Persistent cookie
- **Non-persistent cookie**
  - It is **valid for single session** only. It is removed each time when user closes the browser.
- **Persistent cookie**
  - It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or sign out.

Session Tracking- Using Cookies

- **Advantage of Cookies**
  - Simplest technique of maintaining the state.
  - Cookies are maintained at client side.
- **Disadvantage of Cookies**
  - It will not work if cookie is disabled from the browser.
  - Only textual information can be set in Cookie object.

Session Tracking- Using Cookies

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Session Tracking- Using Cookies

- **How to create Cookie?**

```
Cookie ck=new Cookie("user","sonu");  //creating cookie object
response.addCookie(ck);   //adding cookie in the response
```

Session Tracking- Using Cookies

- **How to delete Cookie?**

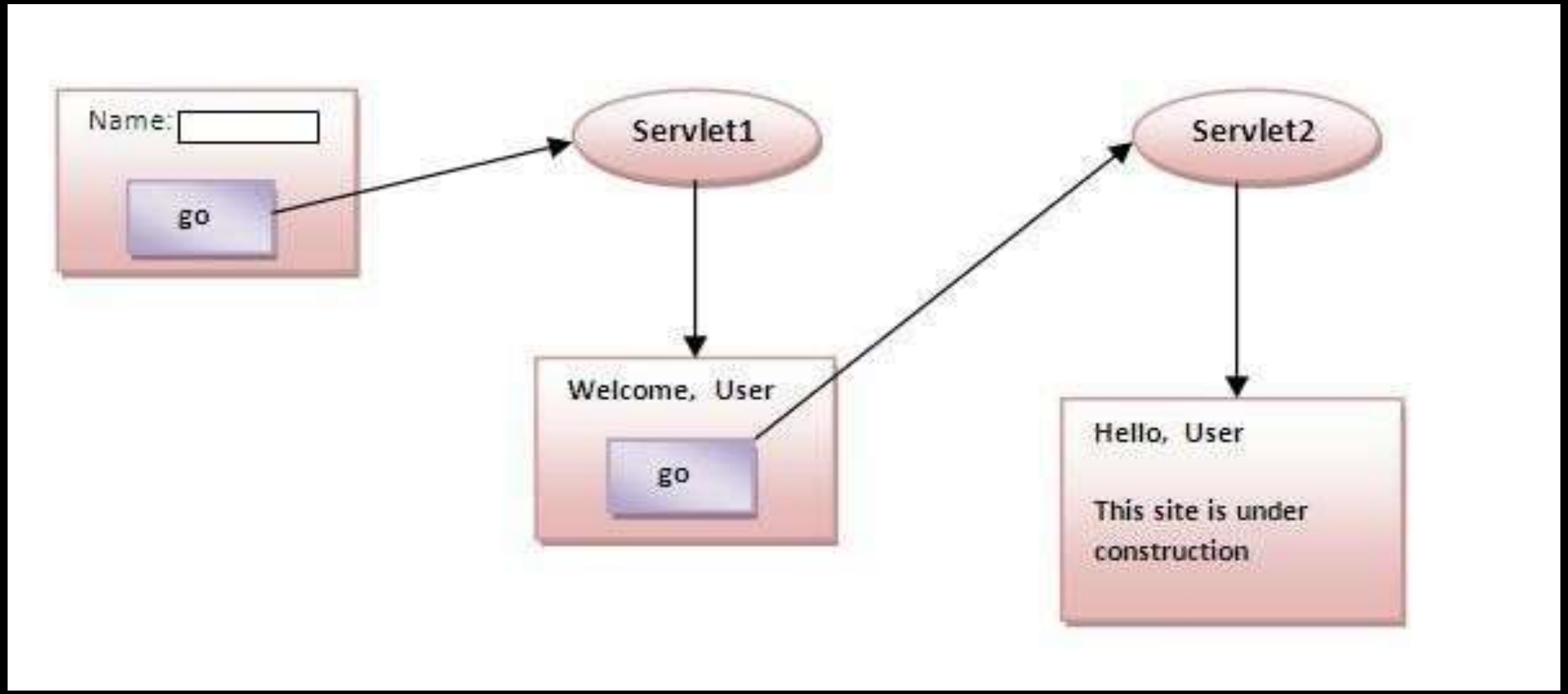
```
Cookie ck=new Cookie("user","") ;//deleting value of cookie
ck.setMaxAge(0); //changing the maximum age to 0 seconds
response.addCookie(ck); //adding cookie in the response
```

Session Tracking- Using Cookies

- **How to get Cookies?**

```
Cookie

ck[]=request.getCookies();

for(int i=0; i<ck.length; i++){

 out.print("<br>"+ck[i].getName()+""+ck[i].getValue()); //printing name and value of cookie

}
```

Session Tracking- Using Cookies Simple example of Servlet Cookies



Session Tracking- Using Cookies Simple example of Servlet Cookies

? index.html

```
<form action="servlet1" method="post">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

Session Tracking- Using Cookies Simple example of Servlet Cookies

Servlet1.java

```
try{
String n=request.getParameter("userName");
out.print("Welcome "+n);
Cookie ck=new Cookie("uname",n); //creating cookie object
response.addCookie(ck); //adding cookie in the response

//creating submit button
out.print("<form action='servlet2'>");
out.print("<input type='submit' value='go'>");
out.print("</form>");
out.close();
}
```

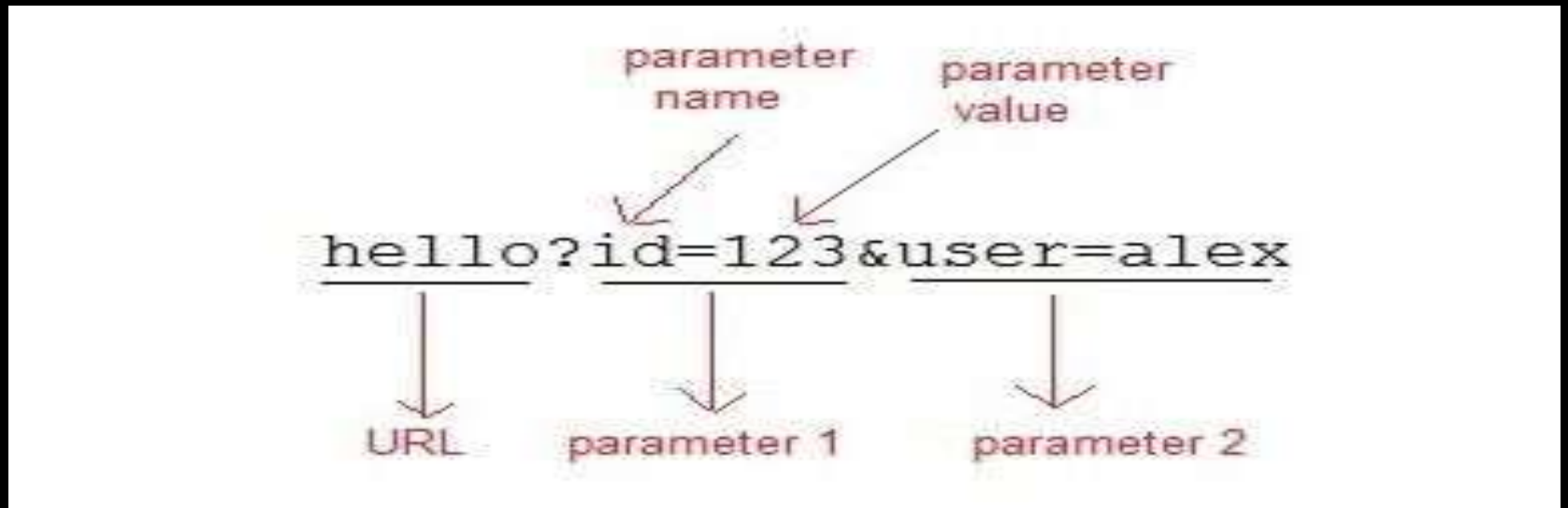
Session Tracking- Using Cookies Simple example of Servlet Cookies

Servlet2.java

```
{
    response.setContentType("text/html");
    PrintWriter out =response.getWriter();

    Cookie ck[]=request.getCookies();
    out.print("Hello "+ck[0].getValue());

    out.close();
    }
```

Session Tracking- URL Rewriting

- If the client has disabled cookies in the browser then session management using cookie wont work.
- In that case **URL Rewriting** can be used as a backup. **URL rewriting** will always work.
- In URL rewriting, a token(parameter) is added at the end of the URL.
- The token consist of name/value pair separated by an equal(=) sign.
- **For Example:**



Session Tracking- URL Rewriting

- When the User clicks on the URL having parameters, the request goes to the **Web Container** with extra bit of information at the end of URL.
- The **Web Container** will fetch the extra part of the requested URL and use it for session management.
- The getParameter() method is used to get the parameter value at the server side.
- **Advantage of URL Rewriting**
  - It will always work whether cookie is disabled or not (browser independent).
  - Extra form submission is not required on each pages.
- **Disadvantage of URL Rewriting**
  - It will work only with links.
  - It can send only textual information.

Session Tracking- URL Rewriting

index.html

```
<form method="post" action="validate">
Name:<input type="text" name="user" /><br/>
 Password:<input type="text" name="pass" ><br/>
 <input type="submit" value="submit">
</form>
```

Session Tracking- URL Rewriting

Validate.java

```
{
    response.setContentType("text/html;charset=UTF-8");
    String name = request.getParameter("user");
    String pass = request.getParameter("pass");
     if(pass.equals("1234")){
     response.sendRedirect("First?user_name="+name+"");
    }
```

Session Tracking- URL Rewriting

First.java

```
{
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String user = request.getParameter("user_name");
    out.println("Welcome "+user);
  }
}
```

Servlet- Data Storage

- Almost all web applications (servlets or related dynamic web server software) store and retrieve data –
  - Typical web app uses a data base management system (DBMS)
  - Another option is to use the file system

Servlet: Concurrancy-

https://docplayer.net/215469659-Data-storage-servlets-and-concurrency.html

One common web application problem: concurrency

Servlet: Concurrancy

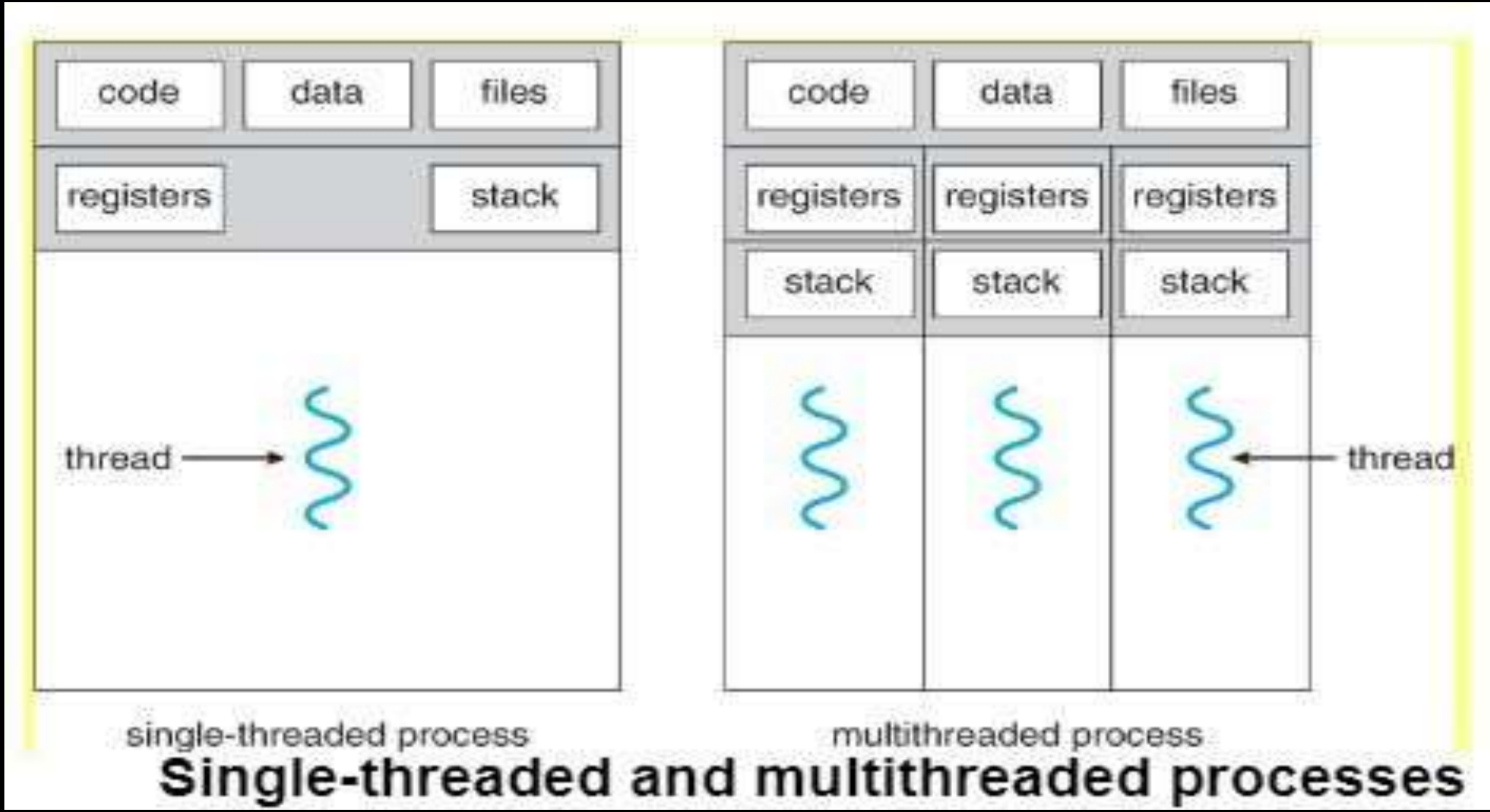
Concurrency means multiple computations are happening at the same time.

- Ex: Web browser loading several images at the same time, or web browser can respond to your mouse clicks while it is still downloading information from a web server are examples of concurrent processing in action on the client side.
- On a server side, multiple requests to the same servlet may be executed at the same time. So concurrency container or web server is multithreaded.

Servlet: Concurrancy

- A thread is a single execution process.
- It is a basic unit of CPU utilization, consisting of own program counter, a stack, and a set of registers.
- A program is multithreaded when multiple threads execute a single instance of a program.

Servlet: Concurrancy



Threading Issues

- Two threads running in Hello Counter concurrently.
- The initial value of visits is assumed to be 17.

User1 Thread	User2 Thread
<started>	
.	
.	
.	
visits++;	
<visits now 18>	
<suspended>	
	<started>
	.
	.
	.
	visits++;
	<visits now 19>
	servletOut.println(...
	visits + ...);
	<outputs 19>
	<completed>
<resumed>	
servletOut.println(...	
visits + ...);	
<outputs 19>	

Thread Synchronization

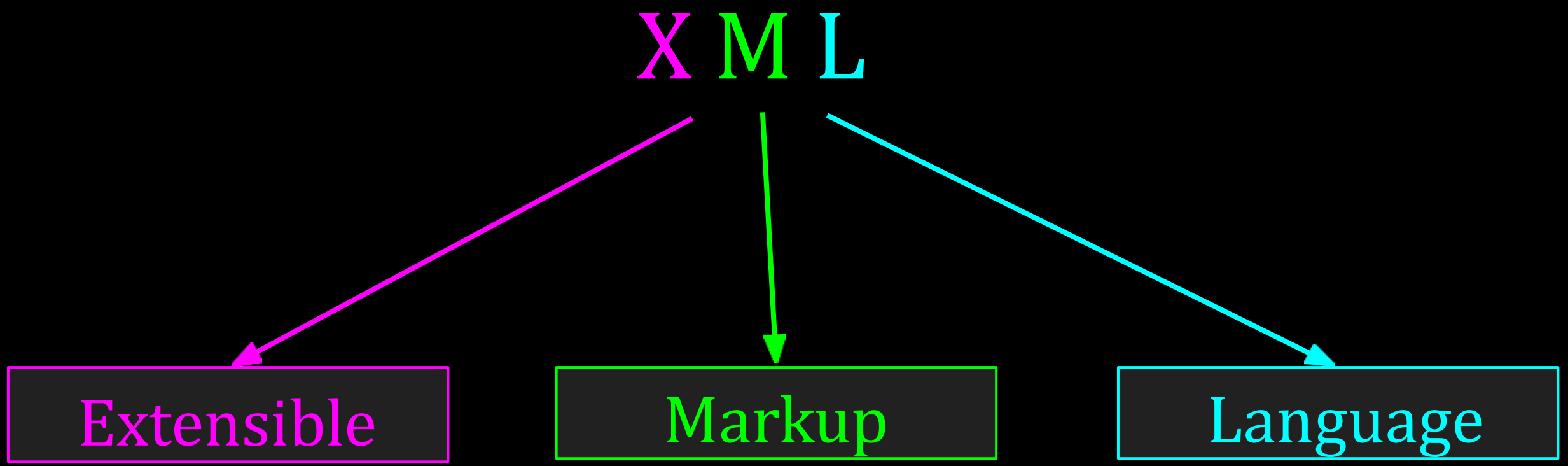
- A servlet must be capable of serving more than one client at a time.
- If several clients issue requests at the same time, methods will serve each client in a different thread.
- service( ), doGet( ), and doPost( ) can handle many concurrent clients.
- It uses lock mechanism to synchronize the threads.

XML

Extensible Markup Language

By Bhavana A. Khivsara

XML



XML

- **Introduction to XML**
  - XML is a software- and hardware-independent tool for storing and transporting data.
- **What is XML?**
  - XML stands for eXtensible Markup Language
  - XML is a markup language much like HTML
  - XML was designed to store and transport data
  - XML was designed to be self-descriptive
  - XML is a W3C Recommendation

XML Does Not DO Anything

- This note is a note to Tove from Jani, stored as XML::
- <note>

  <to>Tove</to>

  <from>Jani</from>

  <heading>Reminder</heading>

  <body>Don't forget me this weekend!</body>

  </note>
- The XML above is quite self-descriptive:
  - It has sender information.
  - It has receiver information
  - It has a heading
  - It has a message body.

XML Properties

XML is a markup language

XML was designed to be self-descriptive

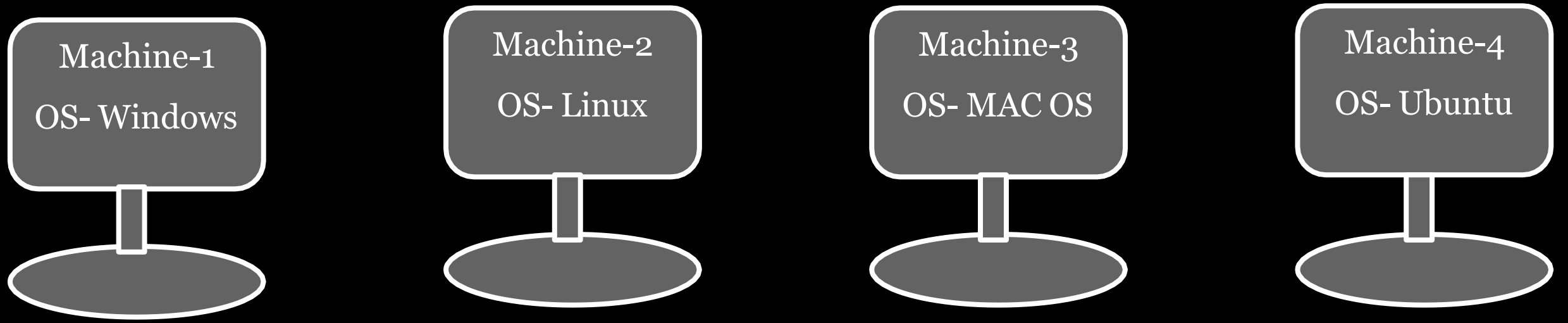
XML is platform & programming language independent.

XML focuses on data

XML was designed to store and transport data

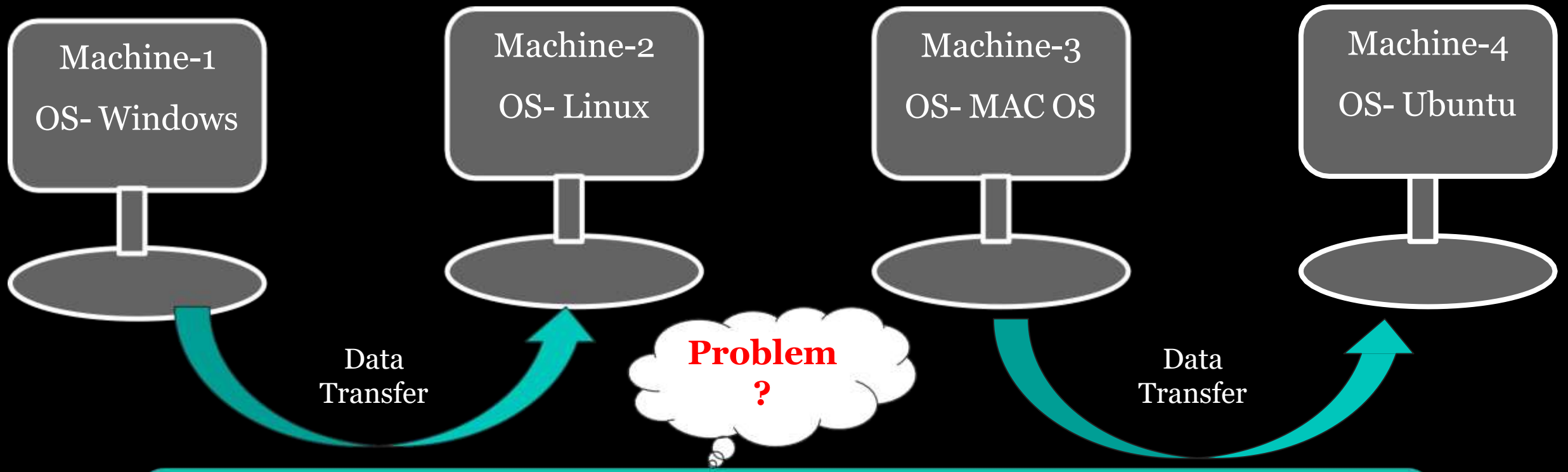
XML is a W3C Recommendation

Why we need XML?



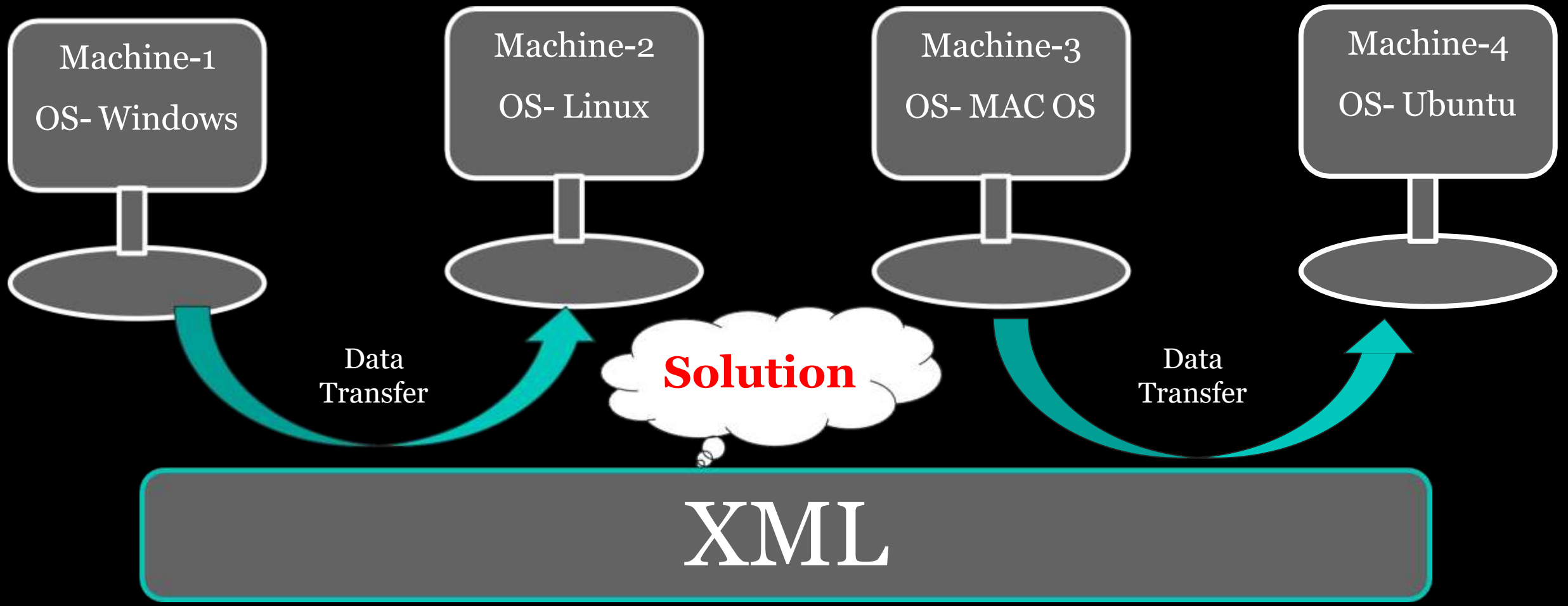
All having different
Operating System & **Data Format**

Why we need XML?

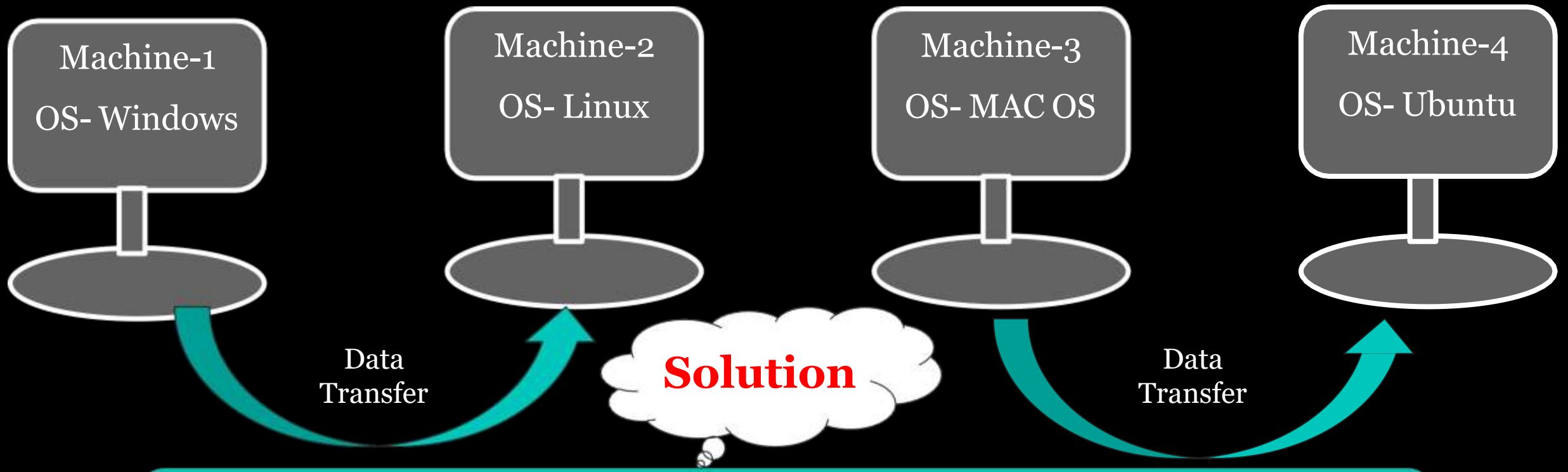


Data Transfer from one machine to another.....
They **need to convert in Compatible Formats**

Why we need XML?



Why we need XML?



With XML, it is so easy to transfer data between such systems as
XML doesn't depend on platform and the language

XML Uses

XML used to store & arrange Data

XML used to exchange the information between organizers and systems

XML use to simplify the creation of a HTML document

XML can easily be merged with style sheets to create desired output.

used for unloading and reloading a database

XML can express any type of XML document

XML Vs HTML

XML	HTML
Is Case Sensitive	Is not Case Sensitive
Has user defined tags	Has its own predefined tags
Used for Transferring Data	Used for Displaying Data
Closing tags are mandatory	Closing tags are not always mandatory
It is Dynamic	It is Static
XML preserve white spaces	HTML does not preserve white spaces

XML Vs HTML

XML	HTML
Used for storing data as structured information	Used for representing content
Strict validation	Loose validation
Defines encoded data	Defines display formatting
No predefined tags or semantics	Predefined tags and semantics
Tags are case-sensitive	Tags are not case-sensitive
New tags/elements can be added by the user	New tags/elements cannot be added by the user

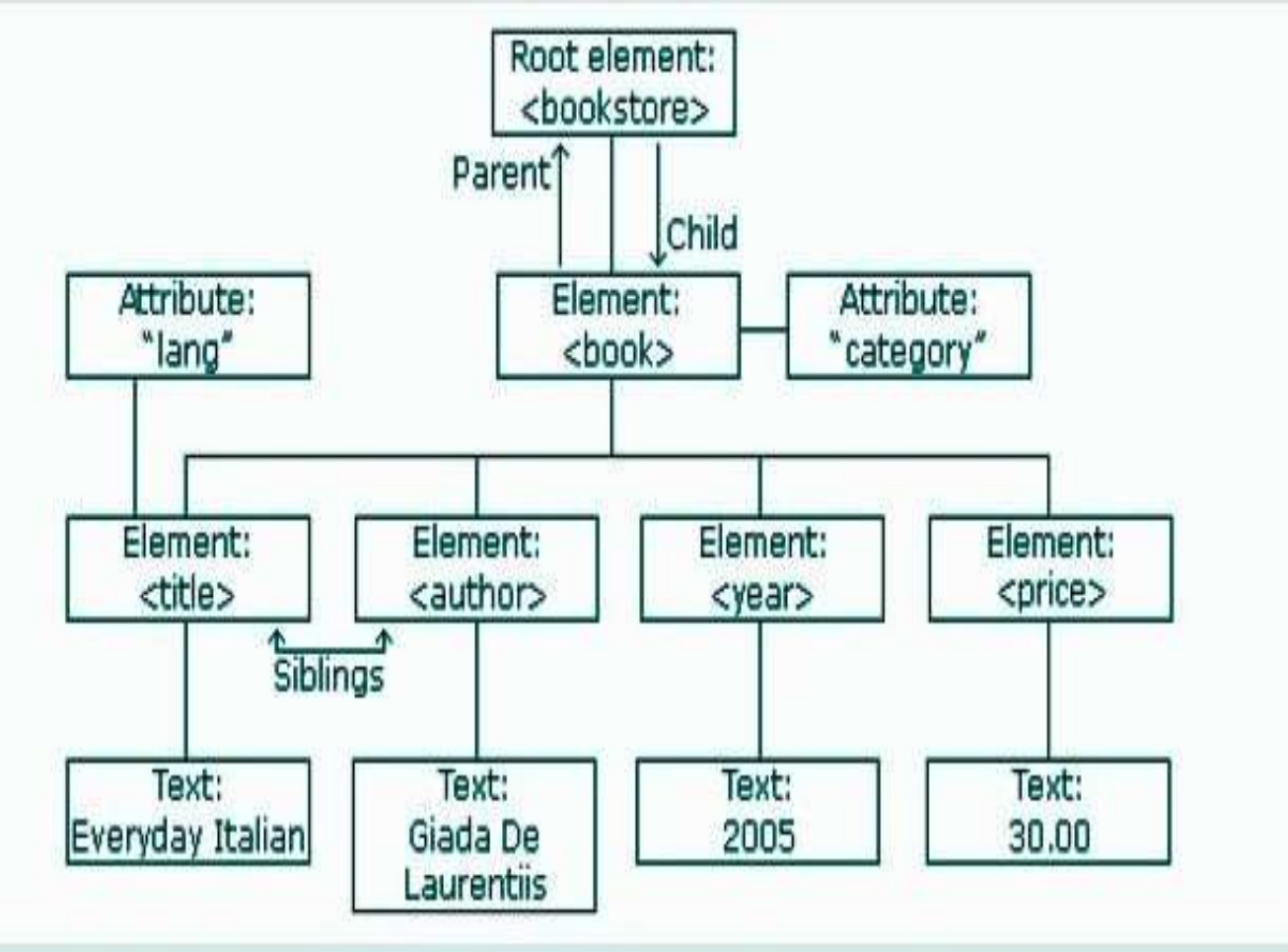
XML Syntax & Example

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Amisha</to>
  <from>Janvi</from>
  <heading>Reminder</heading>
  <body>Don't forget Meeting!</body>
</note>
```

XML Tree Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
 <book category="cooking">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
 </book>
 <book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
 </book>
</bookstore>
```



Practical No-3

XML with CSS

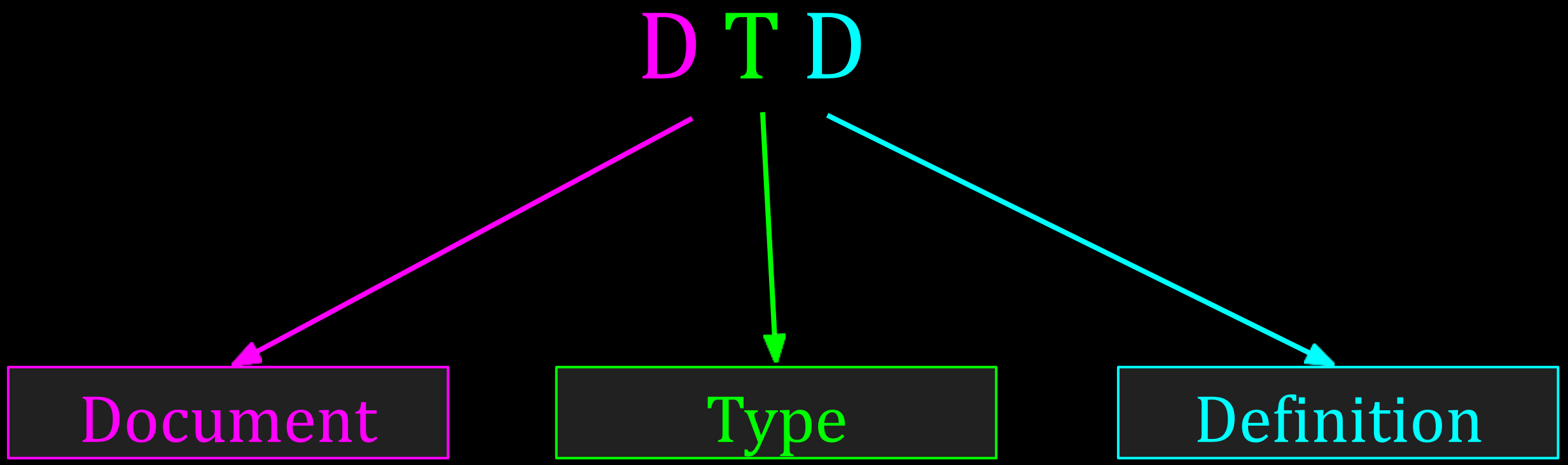
XML with CSS Example

```
<?xml-stylesheet type="text/css" href="filename.css"?>
```

DTD

Document Type Definition

DTD



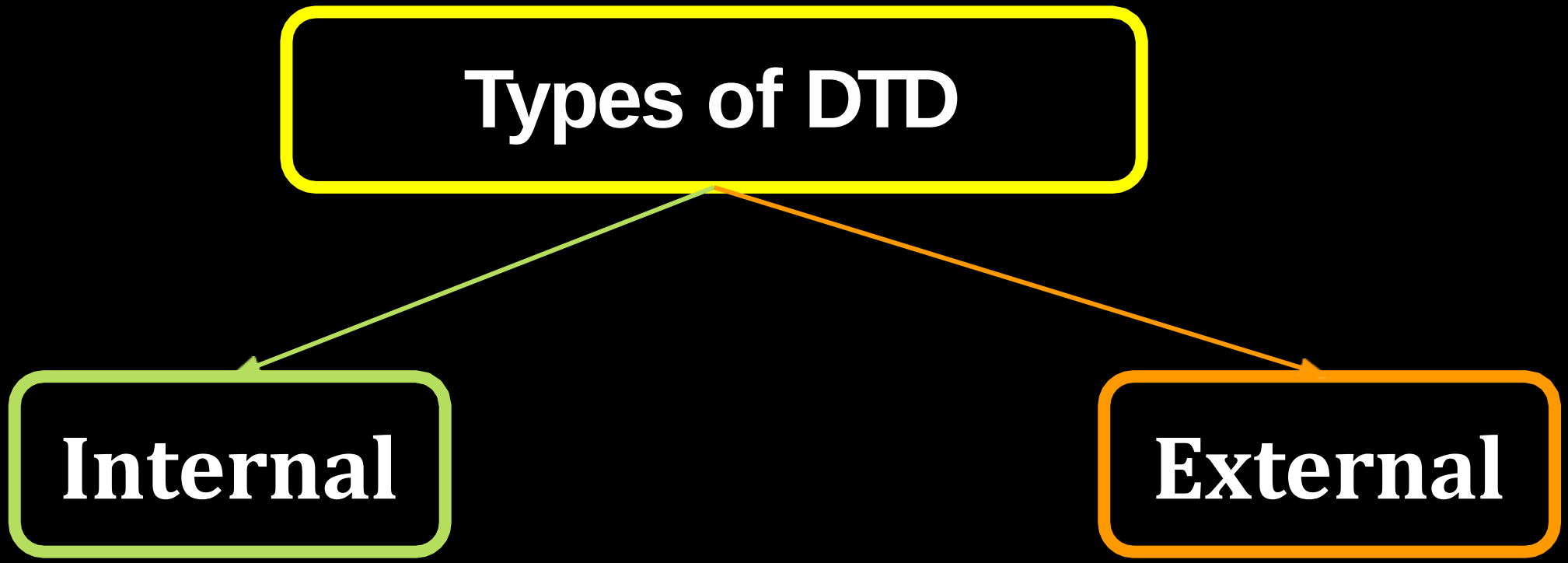
Why TO Use DTD?

A DTD defines the structure and the legal elements and attributes of an XML document

With a DTD, groups of people can agree on a standard DTD for interchanging data.

An application can use a DTD to **verify that XML data is valid**

Types of DTD



Internal DTD– DTD is **declared inside the XML file**, and wrapped inside the <!DOCTYPE> definition

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to> Amisha </to>
<from> Janhvi</from>
<heading> Meeting Reminder</heading>
<body>Today at 5:00PM </body>
</note>
```

XML Prolog

XML Elements

XML Code

DTD Explained

- !DOCTYPE note defines that the root element of this document is note
- !ELEMENT note defines that the note element must contain four elements: "to,from,heading,body"
- !ELEMENT to defines the to element to be of type "#PCDATA"
- !ELEMENT from defines the from element to be of type "#PCDATA"
- !ELEMENT heading defines the heading element to be of type "#PCDATA"
- !ELEMENT body defines the body element to be of type "#PCDATA"

#PCDATA means parsed character data.
The term CDATA, meaning character data

External DTD- Requires 2 Files
1> XML and 2>DTD

XML File

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to> Amisha </to>
<from> Janhvi</from>
<heading> Meeting Reminder </heading>
<body>Today at 5:00PM </body>
</note>
```

DTD File- note.dtd

```
<!ELEMENT note
(to,from,heading,body)>

<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

BUILDING BLOCK OF XML

1. Elements

Elements are the main building blocks of both XML and HTML documents.

Examples of HTML elements are "body" and "table". Examples of XML elements could be "note" and "message". Elements can contain text, other elements, or be empty. Examples of empty HTML elements are "hr", "br" and "img". <body>some text</body>

2. Attributes

Attributes provide extra information about elements.

Attributes are always placed inside the opening tag of an element. Attributes always come in name/value pairs. The following "img" element has additional information about a source file:

The name of the element is "img". The name of the attribute is "src". The value of the attribute is "computer.gif". Since the element itself is empty it is closed by a " /".

BUILDING BLOCK OF XML

3. Entities

Some characters have a special meaning in XML, like the less than sign (<) that defines the start of an XML tag.

Most of you know the HTML entity: " ". This "no-breaking-space" entity is used in HTML to insert an extra space in a document. Entities are expanded when a document is parsed by an XML parser.

The following entities are predefined in XML:

Entity References	Character
<	<
>	>
&	&
"	"
'	'

BUILDING BLOCK OF XML

4. PCDATA

- PCDATA means parsed character data.
- Think of character data as the text found between the start tag and the end tag of an XML element.
- PCDATA is text that WILL be parsed by a parser. The text will be examined by the parser for entities and markup.
- Tags inside the text will be treated as markup and entities will be expanded.
- However, parsed character data should not contain any &, <, or > characters; these need to be represented by the & < and > entities, respectively.

5. CDATA

- CDATA means character data.
- CDATA is text that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

DTD point of view - XML documents

- XML Building blocks for DTD:
  - Elements
  - Attributes
  - Entities
  - PCDATA
  - CDATA

- Example:
  - <book> c </book>
  - <book type="Education"> c </book>
  - <salary> > 5000 </salary>
  - text data (Parsing Character Data)
  - Character Data (> < ‘ “ &)

XML Schema

By Richard A. Rhodes

XML Schema

- **What is XML schema**

XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now a days for example : XSD (XML schema definition).

- An XML schema is used to define the structure of an XML document.
- It is like DTD but provides more control on XML structure.

XML Schema

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which have been validated against Schema.

Visit http://www.xmlvalidation.com to validate the XML file against schema or DTD.

XML Schema

- An XML Schema describes the structure of an XML document. The XML Schema language is also referred to as XML Schema Definition (XSD).
- An XML Schema describes the structure of an XML document, just like a DTD.
- XML Schema is an XML-based alternative to DTD
- XML Schemas are More Powerful than DTD
- XML Schemas are written in XML
- XML Schemas are extensible to additions
- XML Schemas support data types
- XML Schemas support namespaces
- It is primarily used to define the elements, attributes and data types the document can contain.

XML Schema- Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="emp">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="salary" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

XML Schema Example Explained

- <xs:element name="emp"> defines the element called "emp"
- <xs:complexType> the "emp" element is a complex type
- <xs:sequence> the complex type is a sequence of elements
- <xs:element name="name" type="xs:string"> the element "name" is of type string (text)
- <xs:element name="address" type="xs:string"> the element "address" is of type string
- <xs:element name="salary" type="xs:int"> the element "salary" is of type int

DTD VS XML Schema

DTD	XML Schema
It doesn't support namespace.	It supports namespace.
It is comparatively harder than XSD.	It is relatively more simpler than DTD.
It doesn't support datatypes.	It supports datatypes.
SGML syntax is used for DTD.	XML is used for writing XSD.
It is not extensible in nature.	It is extensible in nature.

Standard Generalized Markup Language(SGML)
Extended Markup Language (XML)
Document Type Definition (DTD)
XML Schema Definition (XSD)

XML
Advantages & Limitations(Disadvantages)

Advantages
XML is platform independent and programming language independent, thus it can be used on any system and supports the technology change
It supports Unicode, allowing almost any information in any written human language to be communicated
The data stored and transported using XML can be changed at any point of time without affecting the data presentation
XML allows validation using DTD and Schema. This validation ensures that the XML document is free from any syntax error.
It is XML simplifies data sharing between various systems because of its platform independent nature

XML
Advantages & Limitations(Disadvantages)

Disadvantages

XML syntax is verbose and redundant compared to other text-based data transmission formats such as JSON.

The redundancy in syntax of XML causes higher storage and transportation cost when the volume of data is large.

XML document is less readable compared to other text-based data transmission formats such as JSON.

XML doesn't support array.

XML file sizes are usually very large due to its verbose nature, it is totally dependent on who is writing it.

.JSON : JavaScript Object Notation

JSON is used for storing and transporting data. JSON is often used when data is sent from a server to a web page.

Difference JSON Vs XML :

The JSON format is used to store and transmit data, while XML is used to represent data in a machine-readable way.

XML

Advantages of XSD over DTD :

1. XSD is extensible while DTD is not. This makes it easier to derive new elements from existing elements in XSD.
2. XSD also supports data types, so the content of an element can be restricted. DTD cannot restrict content of an element as it does not support data types.
3. XSD supports element default values, whereas DTD cannot.
4. It is possible to include or import multiple XML schemas within an XML schema. This is not possible with DTD.

AJAX

AJAX

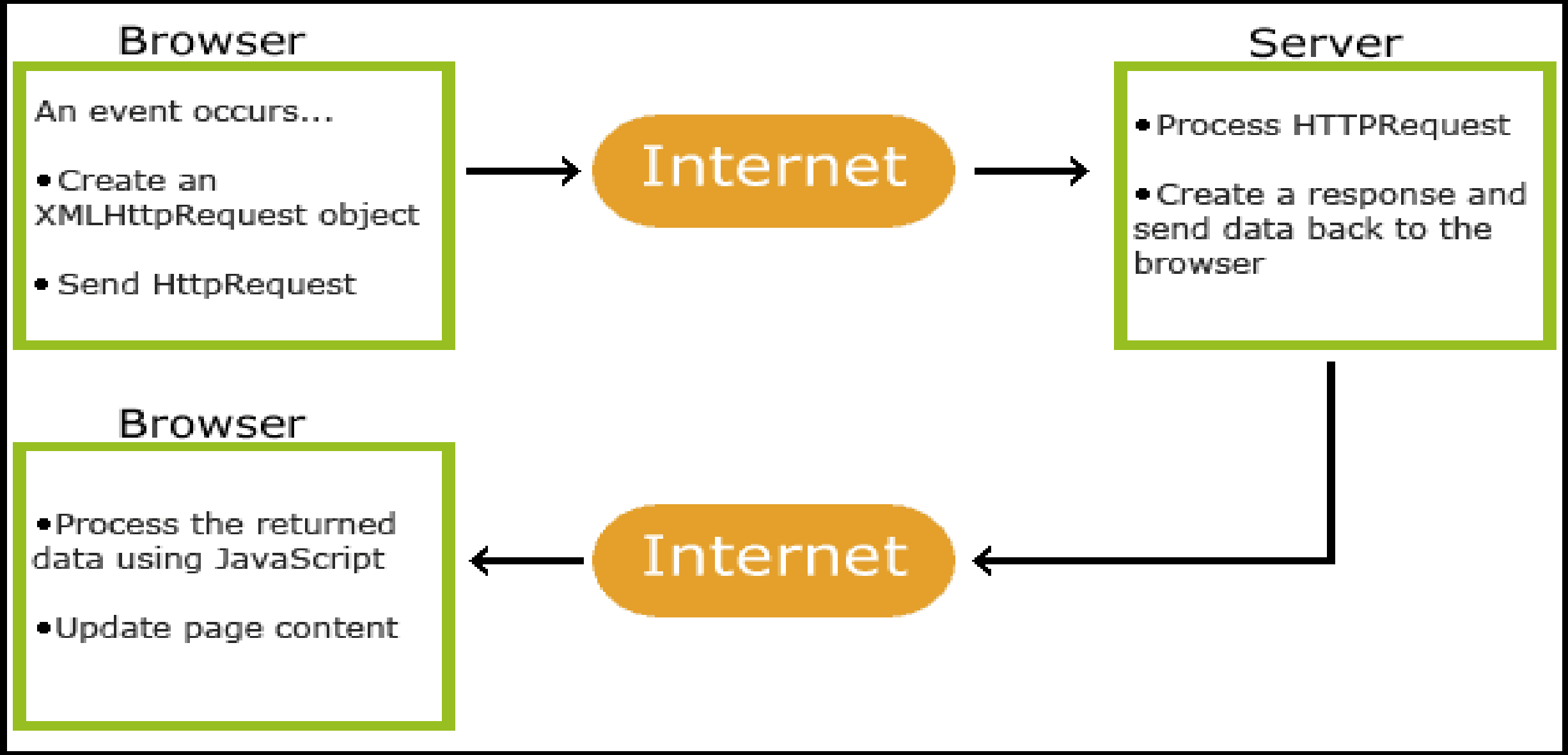
- **What is AJAX?**
  - AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
  - AJAX is not a programming language.
  - AJAX just uses a combination of:
  - A browser built-in XMLHttpRequest object (to request data from a web server)
  - JavaScript and HTML DOM (to display or use the data)

AJAX

- **What is AJAX?**

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page

How AJAX Works?



How AJAX Works?

1. An event occurs in a web page (the page is loaded, a button is clicked)

2. An XMLHttpRequest object is created by JavaScript

3. The XMLHttpRequest object sends a request to a web server

4. The server processes the request

5. The server sends a response back to the web page

6. The response is read by JavaScript

7. Proper action (like page update) is performed by JavaScript

AJAX - Technologies

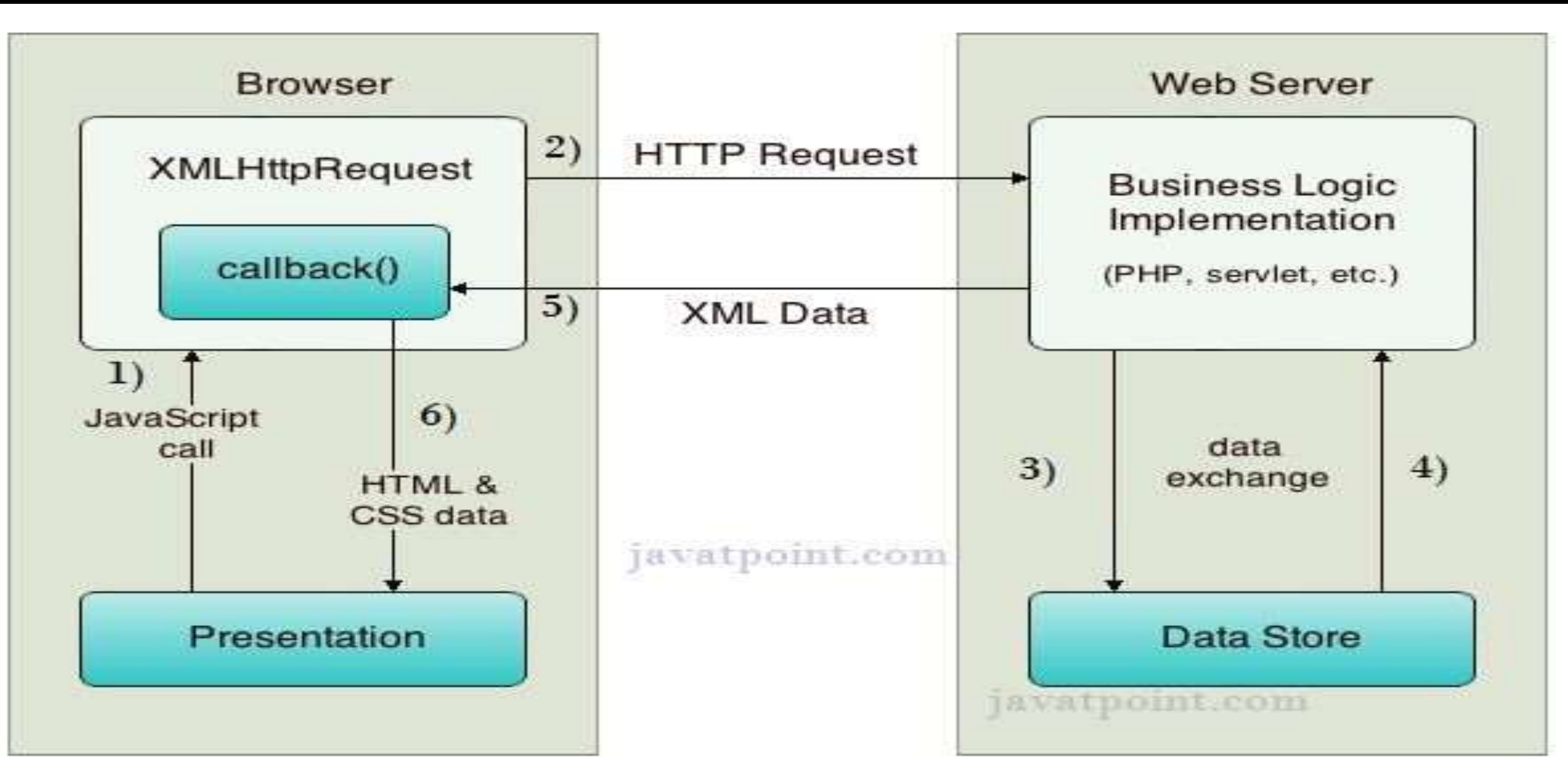
- AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.
- **JavaScript**
  - Loosely typed scripting language.
  - JavaScript function is called when an event occurs in a page.
  - Glue for the whole AJAX operation.
- **DOM**
  - API for accessing and manipulating structured documents.
  - Represents the structure of XML and HTML documents.
- **CSS**
  - Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript
- **XMLHttpRequest**
  - JavaScript object that performs asynchronous interaction with the server.

AJAX – Real Time Examples

Here is a list of some famous web applications that make use of AJAX.

- **Google Maps**
  - A user can drag an entire map by using the mouse, rather than clicking on a button.
- **Google Suggest**
  - As you type, Google offers suggestions. Use the arrow keys to navigate the results.
- **Gmail**
  - Gmail is a webmail built on the idea that emails can be more intuitive, efficient, and useful.
- **Yahoo Maps (new)**
  - Now it's even easier and more fun to get where you're going!

How AJAX Works?



AJAX Processing Steps

- **Steps of AJAX Operation**
  - A client event occurs.
  - An XMLHttpRequest object is created.
  - The XMLHttpRequest object is configured.
  - The XMLHttpRequest object makes an asynchronous request to the Webserver.
  - The Webserver returns the result containing XML document.
  - The XMLHttpRequest object calls the callback() function and processes the result.
  - The HTML DOM is updated.

AJAX Example – table.html

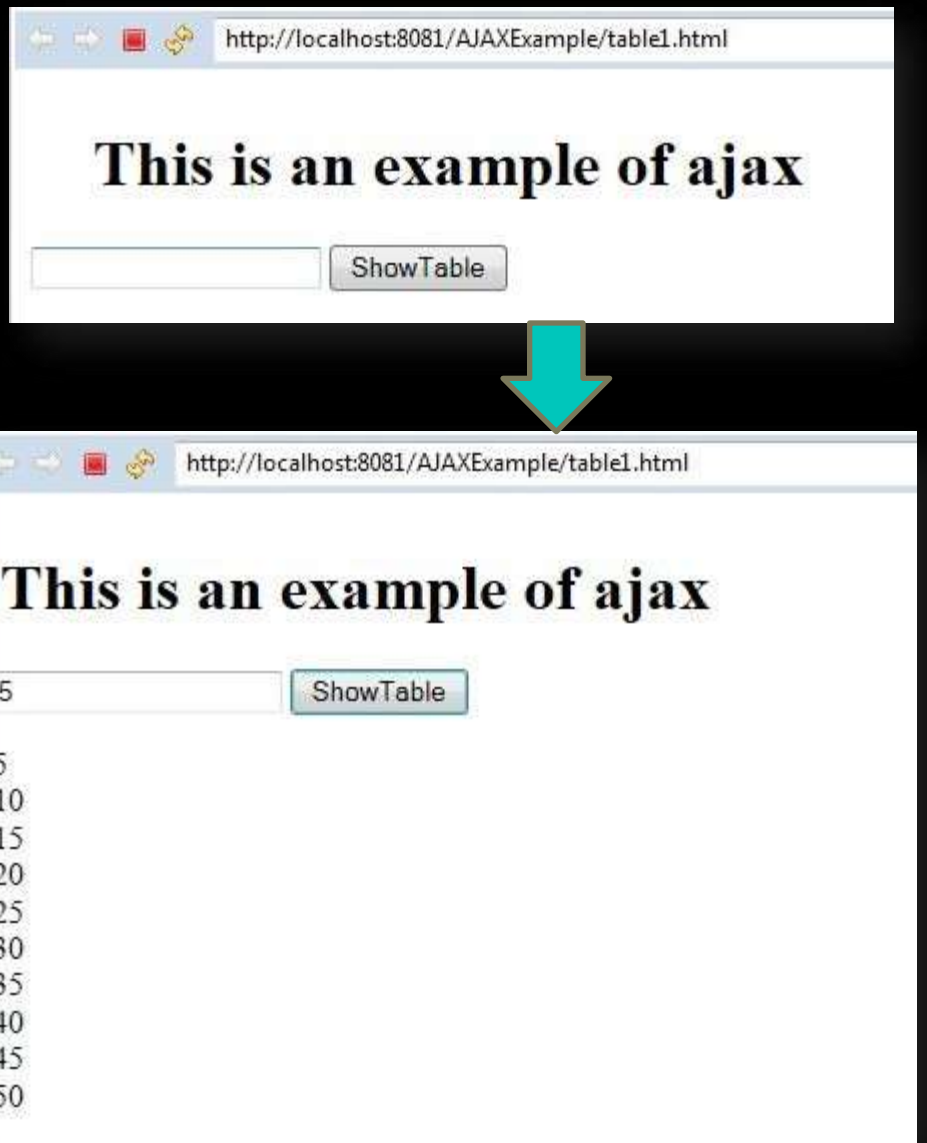
```
<html>
<head>
<script>
var request;

function sendInfo() {
var v=document.f1.t1.value;
var url="index.jsp?val="+v;

if(window.XMLHttpRequest){
   request=new XMLHttpRequest();
  }

request.onreadystatechange=getInfo;
request.open("GET",url,true);
request.send();
}
```

```
 function
 if  request.readyState==4) {
 var val=request.responseText;
document.getElementById('amit').innerHTML=val;
   }
  }
  </script>
 </head>

 <body>
 <h1>This is an example of ajax</h1>
<form name="f1">
<input type="text" name="t1">
<input type="button"
value="ShowTable"
onClick="sendInfo()">
</form>
<span id="amit"> </span>
  </body>
</html>
```

AJAX Example- index.jsp

```
<%
int n=Integer.parseInt(request.getParameter("val"));
 for(int i=1;i<=10;i++)
 out.print(i*n+"<br>");
%>
```

AJAX Example output



References

- https://www.javatpoint.com/cookies-in-servlet
- https://www.javatpoint.com/hidden-form-field-in-session- tracking
- https://www.studytonight.com/servlet/hidden-form-field.php
- https://www.studytonight.com/servlet/url-rewriting-for- session-management.php
- https://www.tutorialspoint.com/jsp/jsp_overview.htm
- https://www.javatpoint.com/jsp-tutorial
- https://www.studytonight.com/jsp/introduction-to-jsp.php
- https://www.studytonight.com/jsp/creating-a-jsp-page.php
- https://www.studytonight.com/jsp/jsp-scripting-element.php
- https://www.javatpoint.com/java-jdbc
- https://www.javatpoint.com/jsp-include-action

Thank You

gharu.anand@gmail.com
Blog : anandgharu.wordpress.com

Prof. Gharu Anand N.