20/11/23 Web

Unit-4

part-A (JSP)

1. Anotomy of JSP page
2. JSP processing
3. Directives
4. Beans with JSP
5. Cookies and session tracking
6. Connecting to a database

part-B (Java Script)

1. Variables
2. Functions
3. Event handling
4. DOM
5. Form Validation

1. ~~Anotomy~~ ~~of~~ JSP page;

* JSP stands for Java Server page
* JSP is a server side scripting language.
* JSP is used to create dynamic web pages.
* A JSP contains both HTML tags & JSP tags.
* JSP is the extended version of Service.

— A JSP page contains 4 types of elements, they are;

1. Expression
2. Scriptlets
3. Directives
4. Declaration.

1. Expression;

Expression element is used to print the output on a screen.

eg; `<%= "JSP" %>`

2. Scriptlets;

— A scriptlets element is used to write a java code

Syn; `<% // %>`

eg; `<% // Java code %>`

3. Directives;

the directives are used for pre-processing (import the packages)

Syntax; ~~Eg~~ `<% @directivename attribute="value" %>`

eg; `<% @page import="java.util.*" %>`

the various directive names are page, include, taglib.

4. Declaration :

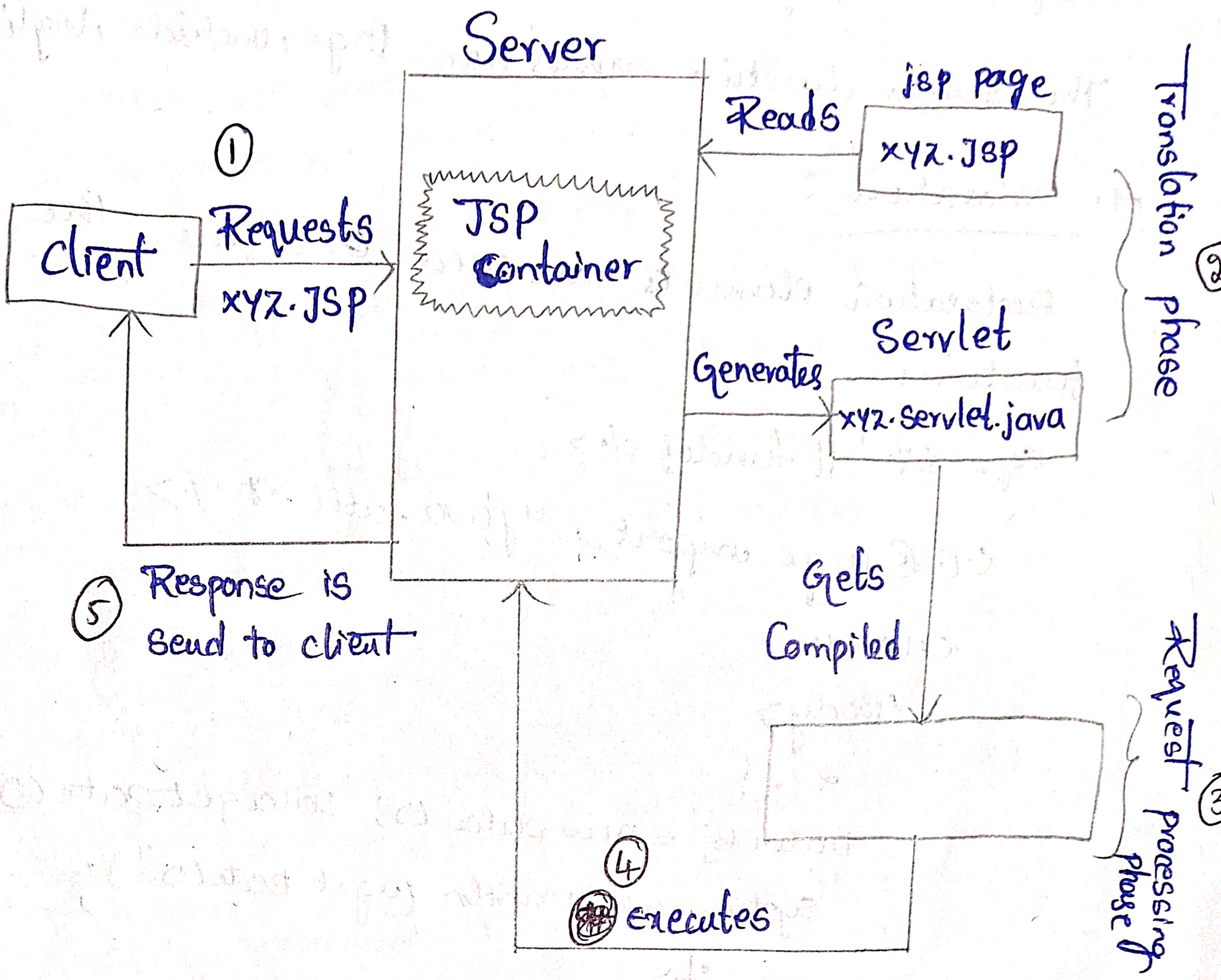
Declaration elements are used to define the functions

eg; <%! //fun def %>

```
<% @page import = "Java.util.*"%>
<HTML>
   <BODY>
      <% !
         Date obj = new Date(); Date get Date();
         System.out.println ("get Date () ");

      %>
   </BODY>
</HTML>
```

2. JSP Processing.



JSP Pages can be Processed using JSP container only Following are the steps that need to be followed while Proccessing the request for JSP page.

1. client makes a request for required JSP page to the Server. The Server must have JSP container so that JSP request can be Proccessed. For instances let the client makes a request the JSP for XYZ.JSP Page.

2. On Receiving this Request the JSP ~~conditioner~~ Container searches and then reads the desired JSP page then this JSP Page is straight away. converted to corresponding servlet. Basically any JSP page is a combination of template text and JSP element. Every template text is translated in to corresponding Println statement.

Every JSP element is converted into corresponding Java code. This phase is called translation phase. The output of translation phase is a servlet. For instance : our XYZ : JSP gets converted to XYZ servlet.java.

3. This servlet is then compiled to generate the servlet class File, using this class the Response can be generated. This phase is called Request Processing phase.

4. The JSP container thus Executes the servlet class file.

5. A Requested Page is then Returned to the client as a Response.

MVC ARCHITECTURE

→ The design model of JSP applications is called MVC Model.

→ The MVC stands for Model - View - controller.

→ The Basic idea in MVC design model is to seperate out design logic into three parts - Modelling, Viewing and controlling.

→ Any server application is classified into three parts such as business logic, Presentation and Request Processing.

→ The Business logic means the coding logic applied for manipulation of application data.

→ The Presentation refers to the code written for look and feel of the web page.

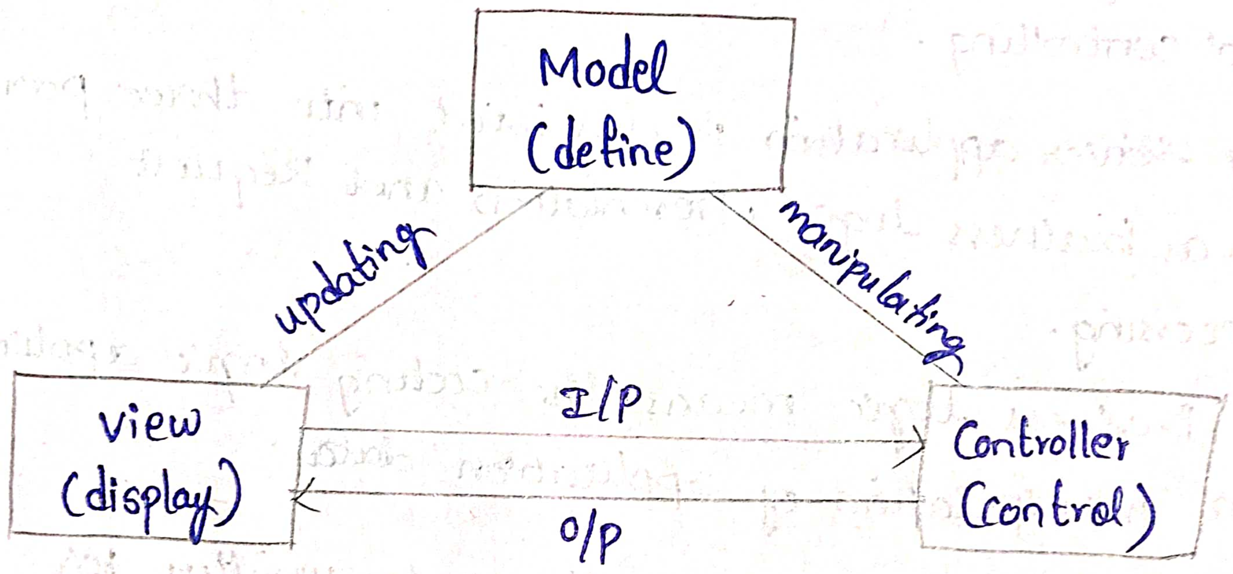
For Example :-

* Background color, Font, style, font size, Placing of controls such as text boxes, Command buttons and so on.

* The Request Proccessing is nothing but a communication of Business logic and Presentations.

* The Request Proccessing is always done in order to generate the Response.

* According to the MVC design model, the Model corresponds to the Business logic, View Corresponds to Presentations and controller Corresponds to Request Proccessing.



4. Beans With Jsp.

Java beans are reusable components. We can use simple java bean in the JSP. This helps us inkeeping the Business logic seperate from the Presentation logic. Beans are used in the JSP Pages as the instance of class. We Must specify the scope of the bean in the JSP page. Here Scope of the bean means the range time span of the Bean for its Existence in JSP. when the bean is Present in Particular Scope its id is also available in that scope.

There are various Scope using which the Bean can be used in JSP Page.

1. Page scope :- The Bean object gets disappeared as soon as the current pages gets discarded. The default scope for a bean in JSP page is a Page scope.

2. Request scope :-
The Bean object remains in Existence as long as the request object is present.

3. Session Scope :-

A session can be defined as the a specific period of a time the user spends browsing the site.

4. Application scope :-

During application scope the bean will get stored to servlet context. Hence particular bean is available to all the servlets in the same web application.

Beans using JSP Example :-

```
Public class Students Bean implements
Java.io.serializable {
   Private string firstName =Null;
   Private string lastName = Null;
   Private int age =0;
   Public StudentsBean () {
   }
   Public string getFirstName() {
      Return FirstName;
   }
    Public string getLastName(){
       Return lastName;
}
    Public int getAge(){
        Return age;
}
    Public void setFirstName(string firstName) {
        this. firstName = firstName ;
}
Public void set LastName (String lastName) {
    this . lastName = lastName ;
}
    Public void setAge (Integer age ) {
        this. age = age ;
    }
}
```

Accessing Java Beans :

```
<html>
<head>
<title> UseBean Example </title>
</head>
<body>
 <jsp :use Bean id = "date "class = "java.util . Date"/>
  <p > the date / time is = <%. = date %>
 </body>
  </html>.
```

5. Cookies & Session Tracking

→ Http is a request - response protocol.

→ Thats means when user wants to access some web page, the web browser makes request to server and server returns that pages as a response

→ But at the same time Http is also called as a stateless Protocol.

→ That Means when browser sends a request to the server & server Proceesses it and sends the Response to the browser and does not Remember anything about the Request.

→ The server should Keep track of the User or the Request made by the user

→ To slove this Problem there are three methods that are Normally used.

1. Use of cookies
2. Embedding Hidden Fields in an Html form.
3. Sending URL String in Response body.

1. Use of Cookies :-

* A cookie is a Name-Value pair information

* The information is passed from server to browser in Response header.

* The browser then returns these cookies unchanged to the server by including the state.

* By returning a cookie to a web server, the browser provides the server a means of connecting the current Page view with previous of page views.

Example for Cookiees :-

```
CookiefirstName = New cookie("First_Name", request.
                    get parameter ("First_name"));

Cookie LastName = New cookie ("Last_name", request.
                    get parameter ("Last_Name"));

FirstName.setMaxAge (60*60*24);
lastName.setMaxage (60*60*24);

Response.addcookie (FirstName);
Response.addcookie (LastName);

%>

<html>

<head>
```

```
<title > setting cookies </title>
</heads>
<body>
  <center>
    <h1> Setting cookies </h1>
    </center>
    <ul>
     <li> <p> <b> FirstName : </b>
       <%=
Request . getparameter (" First -Name ") %>
 </p > </li>
   <li> <p> <b> lastName : </b>
    <% =
Request . getparameter ("Last_Name ") %>
    </p> </li>
    </ul>
   </body>
    </html> .
```

Main.Jsp :-

```
<html>
 <body>
    <form action = "main..Jsp" method = "Get">
First Name : <input type = "text" Name = "first-Name">
   <br/>
 Last Name : <input type = "text" Name = "Last-Name"/>
   <input type = "submit" value = "submit"/>
 </form>
 </body>
  </html>.
```

Session tracking :-

```
<%@ taglib prefix = "c" uri = "http://java.sun.com/
                        Jsp/jstl/core "%>

<html>
<head>
   <title>visitors' counter Demo </title>.
  </head>
  <body>
    <c:set var = "first-cnt" scope = "session"
         value = "${first-cnt + 1}"/>
  <c:set var = " second-cnt " scope = "application"
         value = "${second-cnt + 1}"/>
```

```
<h3> <p> welcome </p> </h3>
The session count of this page is ${first_cnt}.
The application count of this page is ${second_cnt}
</body>
</html>.
```

6. Connecting to a Database in JSP.

* A Database is used to store various types of data.

* To connect a JSP program, first we Need to create a data base and assign a DSN and create tables in that database.

* After creating a table with appropriate Columns with their datatypes and sizes, we can do manipulating for that table using JSP.

* To perform various operations in Databases tables we need to Embed queries in the JSP program.

* Before Performing database operations. The JSP Program must connect to appropriate database with drivers, url, username and Password.

* Queries are to be performed by using <SQL: Query>

SELECT OPERATIONS :-

```
<%@page · import = "java·io·java.util·, Java·sql·*"%>
<%@page import = "javax·servlet·http ·, java·servlet·"%>
<%@ taglib uri = "http://java sun·com/jsp/jstl/core"
        Prefix = "c"%>
<%@ tagliburi = "http://java·sun:com/jsp /jstl/sql"
                 Prefix = "sql"%>

<html>
<head>
<title> select operation </title>
  </head>
<body>
<sql: set DataSource var = "snapshot" driver = "com·
                  mysql·Jdbc·driver"
url = "jdbc : mysql ://localhost /Test"
user = "root" password = "pass 123"/>.

<sql: query datasource = "${snapshot}"
        var = "result">
        SELECT * from Employees ;
        </sql: query>
     <table border = "1" width = "100">
```

```
<tr>
 <th> Emp ID </th>
 <th> FirstName </th>
 <th> LastName </th>
 <th> Age </th>
    </tr>
<c: for Each var= "row" items ="${result.rows}">
<tr>
<td><c: out value = "${row.id}"/></td>
<td><c:out value=" ${row.first}"/></td>
<td><c:out value="${row.last}"/></td>
<td><c:out value= "${row.age}"/></td>
</tr>
</c:forEach>
</table>
</body>
</html>
```

PART-B
(Java Script)

INTRODUCTION

* JavaScript is a scripting language which is used to create interactive web pages.

* Java Script is used along with HTML

* A Javascript code must be written in <script> tag.

  Eg: <script type=" text/javascript">

* javascripts are used to validate the data on the webpage before submitting it to the server.

* javascript can be used to create cookies.

~~Example:~~ comments in Java Script :

* single line comment - //
* Multi line comment - /* - - - - */

Example:

Java script program

```
<html>
<head>
<title> My First javascript </title>
</head>
<body>
<center>
<script type = "text/javascript">
document.write ("Welcome to first page");
</script>
</center>
</body>
</html>
```

VARIABLES

The variables are created in order to show some information. This information can be string or it can be numeric.

Java script program:

```
<html>
<head>
<title> variables </title>
</head>
```

```
<body>
<center>
<script>
 var a,b,c;
  var string;
   a=2;
   b=3;
   c=a+b;
  string = " The result = ";
  document.write (" performing addition of 2 and
                        3." + "<br>");
  document. write (string);
  document. write (c);
 </script>
 </center>
 </body>
 </html>
```

output:

```
         performing addition of 2 and 3

              The " Result = 5
```

CONTROL STRUCTURES

* The various conditional control structures are if, if-else, nested-if, switch-case.

* The various loop control statements are while, do-while, for.

* The Jump control statements are break, continue.

program on javascript (Nested if)

```
<html>
<body>
< ~~#~~ script type = " text / javascript ">
var a,b,c;
 a = 10;  b = 20;  c = 30;
if (a>b)
{
if (a>c)
document.write ("<h3> a is largest numbe
                    </h3>");
else
document.write ("<h3> c is largest numbe
                    </h3>");
}
```

```
else
{
if (b>c)
document.write ("<h3> b is largest number </h3>");
else
document. write ("<h3> c is largest number
                         </h3>");
}
</script>
</body>
</html>
```

output: c is largest number.

~~While~~ ~~st~~ program on javascript (while)

```
<html>
<head>
 <title> Square Value table Converter </title>
</head>
 <body>
 <table border=1 align= "center">
 <th> Number </th> <th> Square </th>
 <script type= "text / javascript">
 ~~x~~  i=1;
      while (i<=10)
      {
      document. write ("<tr><td>"+i+"</td>
                         <td>"+(i*i)+"</td>
                         </tr>");
i++;
}
</script>
</table>
</body>
</html>
```

output:

Number	Square
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

~~pro~~ do- while

The do-while loop is similar to the while loop, the only difference is that the do while executes atleast only once.

program (do-while)

```
<html>
<head>
<title> Demo of do-while </title>
</head>
<body>
<center>
<script type = " text/javascript ">
counter = 1;
do
 {
 document. write ("the statement number:"
                                   + counter);
 document. write ( "<br>");
 counter ++;
 }. while (counter <=5);
</script>
</center>
</body>
</html>
```

output:

```
This statement number : 1
This statement number : 2
This statement number : 3
This statement number : 4
This statement number : 5
```

For LOOP

* This is the most commonly used progra-mming construct.
* The syntax of for loop is

for(initial condition; terminating condition;
                    stepping condition)

<u>program:</u>

```
<html>
<head>
<title>. Square Value table </title>
</head>
<table border=1 align="center">
<th> Number </th> <th> Square </th>
<script type="text/javascript">
 for(i=1; i<=10; i++)
 {
  document.write("<tr> <td>" + i + </td> ~~</tr>~~
                 <td>" +(i*i)+"</td></tr>");
 }
```

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

Output :

Number	Square
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Switch case :

Switch case statement is basically to execute the desired choice.

Program (Switch case)

```
<html>
<body>
<script type="text/javascript">
d=new Date();
ch=d.getMonth();
switch(ch)
{
 case 1: document.write("January");
     break;
 case 2: document.write("February");
     break;
 case 3: document.write("Marcy");
     break;
 case 4: document.write("April");
     break;
 case 5: document.write("May");
     break;
case 6: document.write("June");
    break;
case 7: document.write("July");
    break;
case 8: document.write("August");
    break;
case9: document.write("September");
    break;
case 10: document.write("October");
    break;
case 11: document.write("November");
    break;
case 12: document.write("December");
    break;
}
</script>
</body>
</html>
```

Break: The break statement is used to break the loop.

Program (break)

```
<html>
<head>
<title> Demo of break </title>
</head>
<body>
<center>
<script type="text/javascript">
for (i=10; i> =0 ; i--)
{
  if (i==5)
   break;
}
document.write ("my lucky number is" + i);
</script>
</form>
</center>
</body>
</html>
```

Output:

My Lucky number is 5

Continue:

The continue statement is used in a loop in order to continue(skip)

Program (continue)

```
<html>
<head>
<title> Demo of continue statement </title>
</head>
<body>
<center>
<script type="text/javascript">
for(i=10; i>=0 ; i--)
{
 if (i==5)
 {
  x=i;
  continue;
 }
document.write(i);
document.write("<br>");

}

document.write("The number"+ x +" is missing in
                        above list");

</script>
</form>
</center>
</body>
</html>
```

Functions:-

We can define the function anywhere in the script either in head or body section or in both.

But it is a standard practice to define the function head section and call that function from the body section.

Syntax:-

```
function name_of_function(arg1, arg2, ... argn)
{
   ...
   Statements
}
```

Program

```
<html>
<head>
<script type="text/javascript">
function my_fun()
{
 document.write("I am in function");
}
</script>
</head>
<body>
<center>
<script type="text/javascript">
document.write("This statement is before a function
                    call");
document.write("<br>");
my_fun();
</script>
</form>
</center>
</body>
</html>
```

Output:

```
This    statement    is before    a function call

        I am in function
```

Program: (with parameters)

```
<html>
    <head>
        <script type="text/javascript">
            function fun1(val1, val2)
            {
                x = val1 + val2;
                document.write(x);
            }
        </script>
    </head>
    <body>
        <script type="text/javascript">
            fun1(5, 6);
        </script>
    </body>
</html>
```

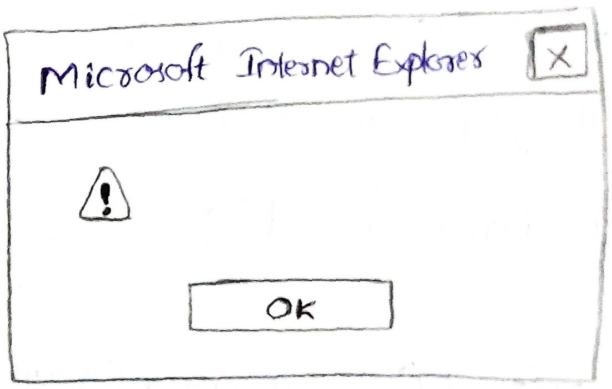
Outputs

11

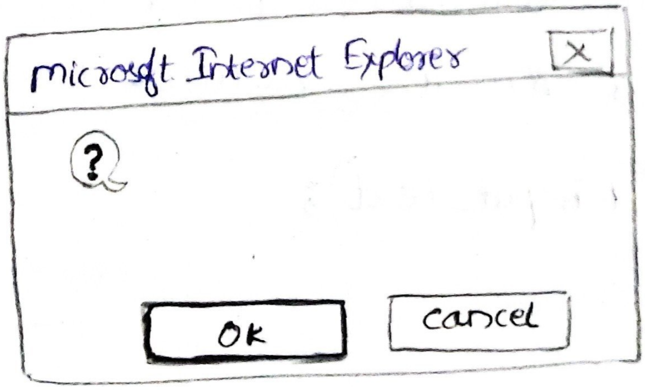
* Event Handling

→ Pop up Boxes

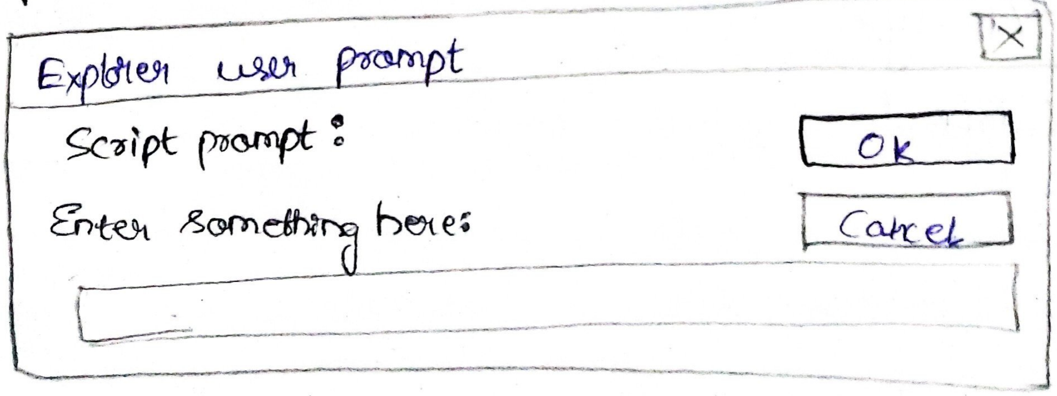
* Alert box:-



* Confirm box:-



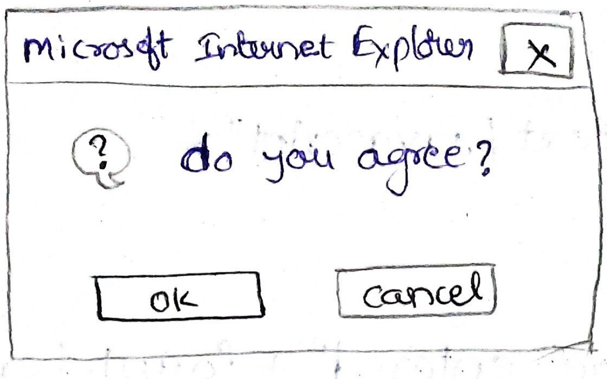
* Prompt box:



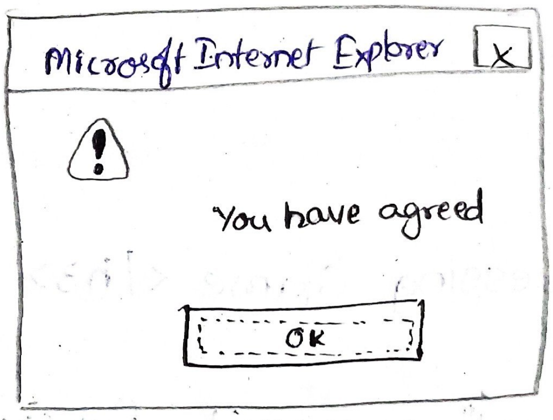
Program:-

```
<html>
<head>
<title> Introduction to pop up box </title>
</head>
<body>
<center>
<script type="text/JavaScript">
if(confirm("do you agree ?"))
 alert("You have agreed");
else
input_text = prompt("Enter your condition here...","");
/* the value entered in prompt box is returned
and stored in the variable text */
alert("Hi" + input_text);
</script>
</center>
</body>
</html>
```

Outputs



do you agree?



You have agreed

Events

* An Event is a change of state of an object in HTML.
* Events represent an activity performed by user/browser
* Java script handles the Events by using Event handlers
* The various events & event handlers are

Mouse Events

click	→ Onclick
mouseover	→ onmouseover
mouseout	→ onmouseout
mousedown	→ onmousedown
mouseup	→ onmouseup
mousemove	→ onmousemove

Keyboard Events

keydown	→ onkeydown
keyup	→ onkeyup

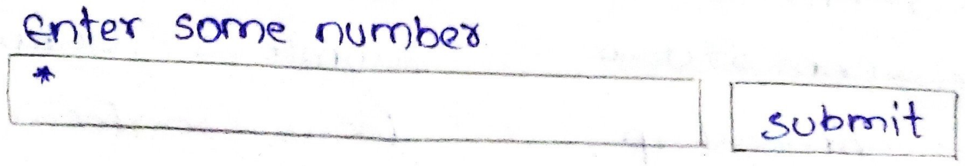
Form Events

submit	→ onsubmit
focus	→ onfocus
load	→ onload

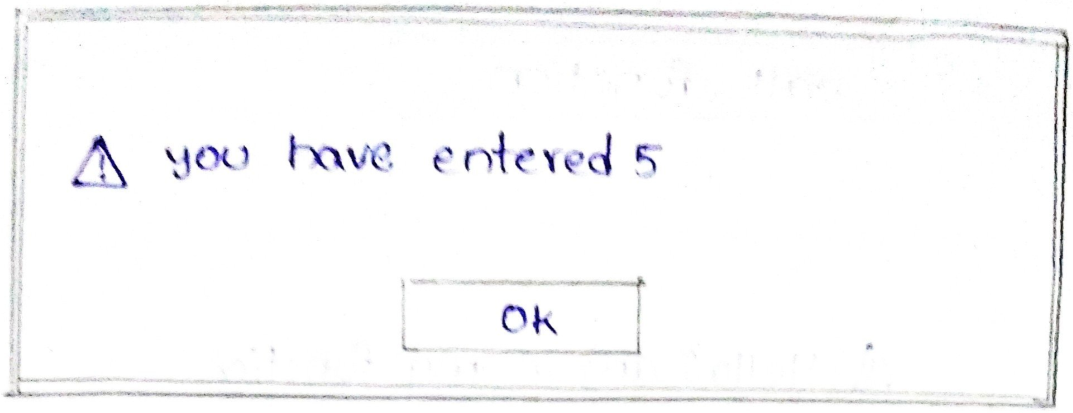
program

```
<html>
<head>
<script type="text/javascript">
 function func()
 {
 alert("you have entered" + form1.text1.value);
 }
</script>
</head>
<body>
<center>
<h3> Number Guessing Game </h3>
</center>
<form name="form1" onsubmit=func();>
Enter some number <br>
<input type="password" name="text1">
<input type="submit" value="submit">
</form>
</body>
</html>
```

output:



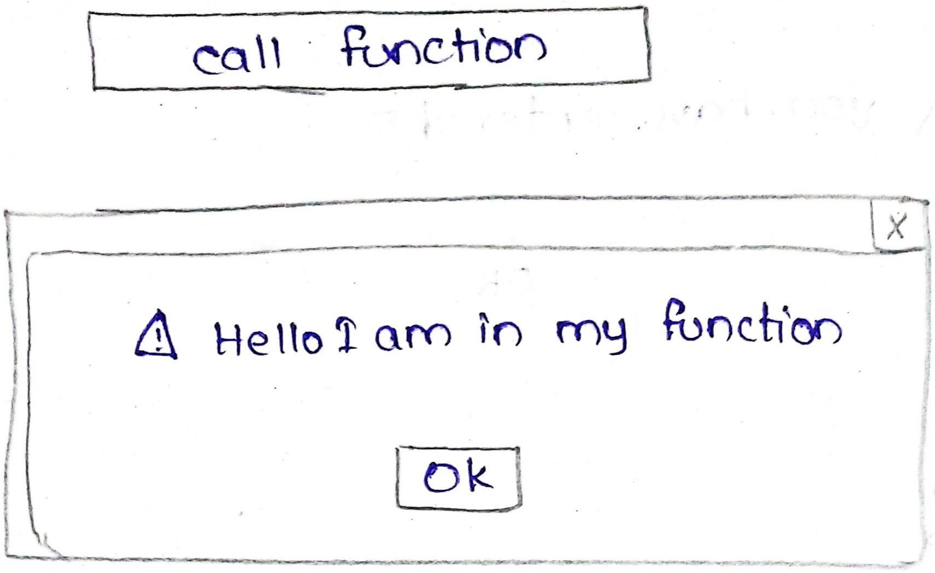
```
enter some number
*                                   submit
```


⚠ you have entered 5

program

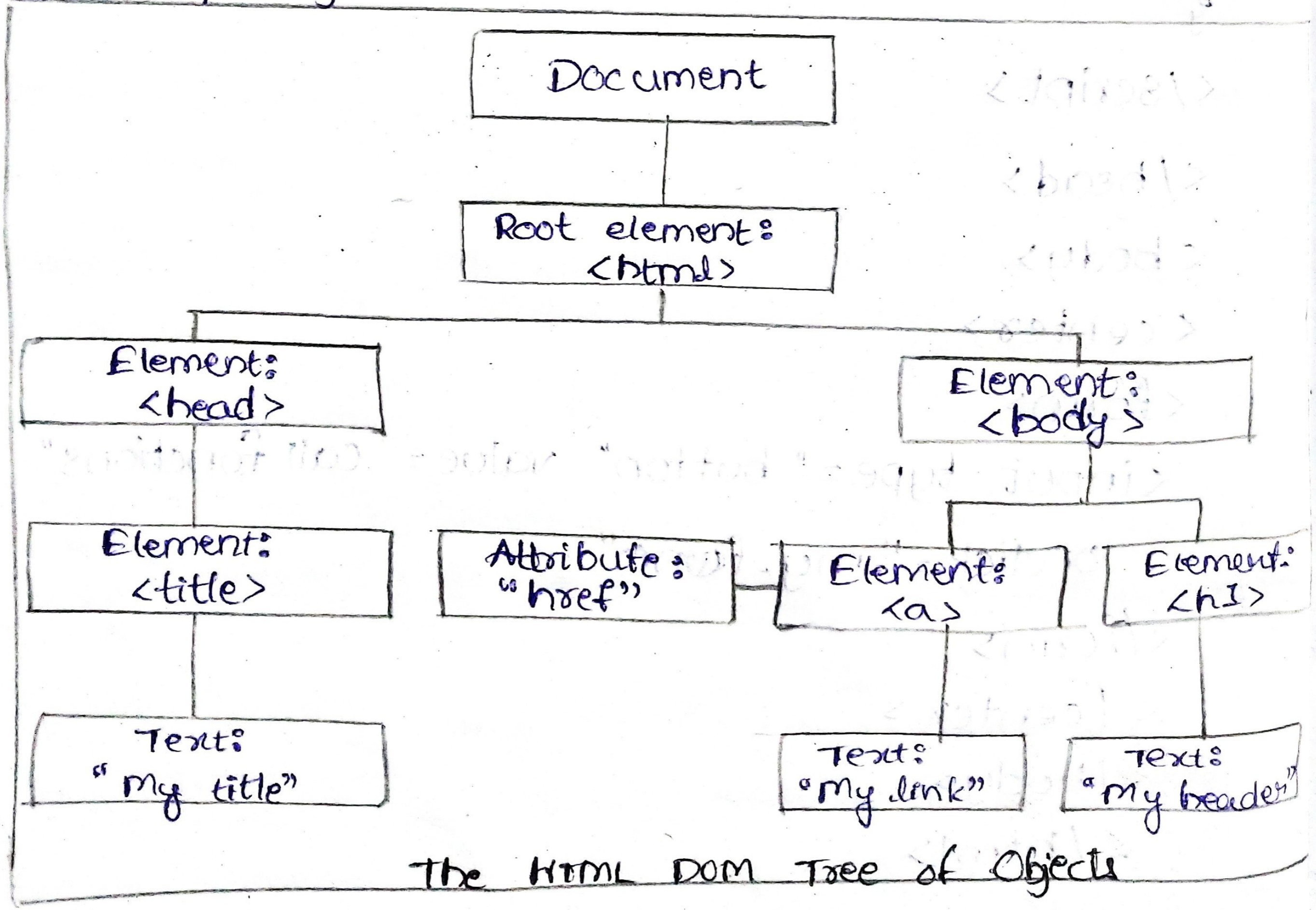
```
<html>
<head>
<script type = "text/javascript">
Function my-fun()
{
alert (" Hello I am in my function");
}
</script>
</head>
<body>
<center>
<form>
<input type=" button" value= "call functions"
    onclick=" my-fun()">
</form>
</center>
</body>
</html>
```

Output:



Document Object model:

* When a web page is loaded, the browser creates a Document Object model of the page.
* The HTML DOM model is constructed as a tree of objects:



The HTML DOM Tree of Objects

The benefits of object model in javascript are

1. It can change all the HTML elements in the page.
2. It can change all the HTML attributes in the page.
3. It can change all the CSS styles in the page.
4. It can remove existing HTML elements and attributes and add new of them.
5. It can react to all existing HTML events in the page and can create new of them.

* Form Validation:-

Validation is needed to validate the data used by the user on a form

Javascript is used to validate the form at client-side.

Validation at client-side is more faster than in server-side.

We can validate usernames, passwords, emails and other formats.

Eg: Program (name & password Validation)

```
<script>
function validateform() {
var name= document.myform.name.value;
var password =document.myform.password.value;

if(name==null || name=="") {
  alert("Name can't be blank");
  return false;
} else if(password.length<6) {
  alert("Password must be at least 6 characters
          long.");
  return   false;
 }
}
</script>
<body>
<form name="myform" method="post" action="abc.jsp"
   onsubmit="return validateform()">
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password">
<br/>
```

```
<input type="submit" value="register">
</form>
```