Assignment- 1

1.

***Update the Doctype**

Ensure your HTML documents start with HTML 5 doctype to leverage HTML5 features:

```
<!DOCTYPE html>
```

***Use New Semantic Elements:**

Replace generic <div> and <span> tags with HTML5 semantic elements to improve readability & accessibility

* <header> for headers

* <nav> for navigation Skills

*<Section> for Sections

* <article> for Self - contained content

* <aside> for Side content

* <footer> for footers

* <main> for primary content

***update forms with new input**

HTML5 introduces new input types that enhance user experience and validation:

* <input type = "email" > for email address

* <input type = "url" > for URLs

* <input type : "date" > for dates

* <input type = "number"> for numeric input

***Add form Attributes**

Utilize new attributes to improve from functionality

* required to make fields mandatory

* placeholders ~~to~~ improve hint text

* pattern for Custom validation with regex

*auto complete to suggest or remember inputs

***Implement new form elements:**

Take advantage of new form elements:

* < datalist > to provide options for an <input>

***Improve Accessability**

*ARIA roles and attributes : use ARIA roles and attributes to improve accessability.

Example for transitioned form

```
<form action = "/submit"
method = "post">
  <div>
    <label
for : "username" > Username : </label>
        <input type : "text"
id : "username" name : "username"
placeholder : "Enter your username"
required >
</div>
<div>
<input type : "Submit"
value : "Submit" >
</div>
</form>
```

2.

HTM

2. HTML Structure:

```
<!DOCTYPE html>
<html lang = "en" >
<head >
   <meta charset = "UTF-8" >
   <meta name = "viewport"
content = "width = device - width, initial - scale = 1.0" >
     <title> user Registration from <ltitle>
     <Style >
      body {
         font - family : Arial
   Sans. serep;
           max - width : 680 Px ;
           margin : auto ;
         }
         form {
            display : flex ;
            flex - direction :
   column;
      }
      label {
         margin : 10Px 0 5px;
    }
    input text area {
        padding : 8 px
    margin - bottom : 10px ;
    border : 1Px solid 11 ccc;
   }
   button {
      Padding : 10px ;
      border : none;
      border - radius : 4px ;
      back - ground - color;
  #4CA 50;
       Color : white ;
       Cursor: pointer ;
```

```
#45a049;
  }
  </Style>
  <h1> user Registration </h1>
  <formed ="registration Form" >
     <label for "name"> name: </label>
  document. get Element By Id (registration form'). add
  Event listener ('Submet; function (event) {
     varname : document . get Element By ID
   if (!name || ! email || ! message) {
      alert ("All fields are required");
      }
    </script>
    </body>
    </html>
```

3.

*** Fixed Layout**

```
  <!DOCTYPE html>
  <html  lang : "en">
  <head>
    <meta  charset  = "UTF -8">
    <meta  name  ="viewport"
  Content : "width = device - width, initial - scales = 1-0 >
   <Style>
    body {
        margin : 0;
        padding : 0;
     <div  class= "Container">
       <div  class ="header"> fixed layout
          Header </div>
       <div  class = "main" > fixed width
          content </div>
       </body>
       </html>
```

* Fluid lay

By incorp

fixed , fl

4.
Fluid (

Pe

* Fluid layout

```
<!DOCTYPE html>
<html lang = "en">
<head>
  <meta charset = "UTF-8">
  <meta name = "view port">
}
main {
  Padding : 20px;
}
</Style>
</head>
< body>
  < div class = "container" >
  <div class = "head"> Fluid layout Header
  </div>
</body>
</html>
```

By incorporating these css techniques we can create website with fixed, fluid and responsive layouts

4. Fluid Grids:

Percentage based widths

```
<!DOCTYPE html>
<html lang = "en" >
<head >
<meta Charset = "UTF-8">
<meta chas = "viewport" >
Content = "width device-width
initial Scale = 1.0" >
<title> Responsive layout with fluid creds </title>
<style>
    body {
        margion : 0;
        font-famely = Arial;
grid-template-columns: 1fr;
}
}
<div class = "Container">
  <div class ="item"> Content1</div>
  <div class = "item"> content 4 </div>
 </div>
 </body>
 </html>
```

Flexible Images

```
<!DOCTYPE html>
<html lang: "en" >
<head>
  <meta charset : "UTF-8">
  <meta name : "viewport"
Content : "width" = device-width, initial-scale = 1.0">
</Style>
</head>
<body>
  <img src : "example.jpg"
 alt : "Example Image " >
 </body>
 </html>
```

use fluid grids for flexibility media for responsive design and techniques like flexible images and touch friendly elements to enhance usability and performs across device

5. Webpage Design: Product listing:

Sample HTML Layout that it includes product images, description and pricing, laid out with tables and list where appropriate

HTML Structure

```
<!DOCTYPE html>
<html lang = "en">
<head>
    <meta charset = "UTF -8">
    <meta name = "viewport"
  Content = "width. device -width , initial -scale >
    <title> Product listening </title>
   <Script, src = Srcipt .js </ Srcipt >
   </head>
   <body>
       <header>
           <nav>
               < ul>
                   < li>< a
   href : " # contact "> contact </a></li>
               </ul>
               </nav>
             </header>
             <main>
            < footer>
             < p> © copy ; 2024 My company </p>
            </footer>
      < body>
     </html>
```

Java Script (script.Js)

```
document.add Event Listener ("Dom content loaded ()=>{
    Console. log (" Document loaded and Parsed");
});
```

Assignment - 3

1.

```
import
javax.servlet. servlet Exception;
import
javax. Servlet,annotation . webservlet;
import
javax. Servlet. http. Http Servlet;
import  Javax. Servlet .http . Http Session;
import  Java. io. exception ;
import  Java. io. Printwriter.
Public class  Track session servlet extends
        Http  Servlet  {
      @ override
      Protected  void
doGet  (Http Servlet Request  request, Http Servlet Response response)
throws Servlet Exception,
IO Exception {
  response. Set content type ("text/html");
  Http session  Session = request. get session.
  Integer  access count = (Integer)
  Session .get Attribute (" access count ");
     if  (access count == null) {
         access count=0;
  String sessionId = session. get Id ();
  long creation time = long last creation time ();
  print writer out = response. Get writer ();
  out. println (" <html> <body >");
   out. println ("<h2> session tracking </h2>)
  ID = " + session Id + " </P>").
  Time : " +new
  Access : " + access count + " </P>");
  out. println (" </body></html>");
}
 }
```

output :
<P> S
<P>
<P>
<P>

2. Scen

In

list

Th

separ

Solutio

imp
imp
imp
im
im
im
im
Pu
Pr

do
P
P
F
}

}

Outp

nse response)

Output :
```
<p> session ID : 1234567890abc </p>
<p> Creation time : Mon Sep 10 15 : 20:30 2024 </p>
<p> Last accessed time : Mon Sep 10 15 : 25: 10 2024 </p>
<p> Number of Acesses : 5 </p>
```

2. Scenario using JSTL :

In a web application you have requirement to display a list of products with varying categories
The product list needs to be dynamically categorized into separate sessions on web page

Solution using JSTL
```
import javax. servlet. ServletException;
import javax. servlet. annotation. web Servlet;
import javax. servlet. http. Http Servlet ;
import java. io. IO Exception
import Java. util. List;
import. Java. util. map ;
import Java. util. HashMap ;
Public class
product list servlet extends Http servlet {
    Protected void
doGet (Http servlet request HTTP Servlet)
product ("Shirt", "clothing", 30, true),
product ("washing machine", "Home appliance, false)
Product ("Shirt", "clothing", 30, true),
);
product list . JSP "), forward request, respon se
}
}
```
Output:
```
<Strong laptop </Strong> - $1000 - Available
<clothing> Shirt - $30 - Available
<Home appliance> washing machine - $30 - Out of stock)
```

3.
To achieve this setup HTML page Add Java Script for automatic refresh and confirm dialog

HTML and Java Script code:

```
<! DOCTYPE html>
< html lang = "en" >
<head >
  <meta Charset = "UTF-8" >
  <meta name = "view port">
Content = "widm = device -width, initial - scale = 1.0" >
  <title> Stock market Quotes > / < title>
< Script>
     var user response confirm the page will refresh in
       20 seconds do you need more time ?"
   if  (!user response ) {
      window. location .reload ()
     }. refresh Interval );
<h1 > Stock market Quotes </h1>
<P> Here you can display real- time
   Stock market quotes . </P>
  </body>
  </html>
```

Output:

Page Display.

The Page will refresh in 20 seconds Do you need more time !

(Cancel) (ok)

4. Steps to Interrogate a payment gateway for Java

```
import your. packge .name. Payment service ;
Import package . name payment service port type;
Public class payment client {
     Public static void main (String [] args)
          Payment service service = new payment service ();

     String response = port process payment
```

("amount", "currency", "payment details");

System.out.println ("Response:"+response)
}
}

output

Payment response : Payment Successful for amount 100.00 USD

Assignment - 4

1. Configure Data source in context.xml

```
<context>
    <Resource name = "jdbc/mydata source"
  type = "Javax. sql. DataSource"
                    maxTotal = "20"
                    maxIdle = "10"
    maxwaitmillis = "10000"
    username = "dbuser"
    password : "dbpassword"
    driver name: "com. mysql. cj. jdbc. Driver"
    url = "Jdbc:mysql : // local host : 3306 /my database
    />
    </context>
```

Collable Statement

```
import java.sql. collable. Statement;
import java.sql. connection;
import java.sql. SQLException;
import java.sql. Types;
Public class collable statement Example {
Public void callstored procedure (int employee id)
        String sql "{call getEmployee Name (?,?)}";
  try (connection conn. Database utility
                connection ());
     employee name = stmt. get String (2);
     System. out. println ("Employee Name:" + employee name);
     } catch (SQL Exception) {
          e. print Stack Trace ();
     }
   }
}
```

Output.

```
Employee ID: 1, Name: John
Employee ID: 2, Name: Jane
Employee ID: 3, Name: Emely
Rows updated : 1
Employee name: John
```

2. Life cycle of phases of a JSP page:

1. Translation phase:
   * The JSP page is translated into a Java servlet by JSP engine (e.g. Html mixed with JSP tags)
   * This phase ensures that JSP content is converted int a form that Java servlet contance can execute.

2. Compilation phase:
   * The Java Source code generated from translation phase
   * compilation ensures JSP is converted into executable

3. Initialization Phase
   * The Servlet contains initializing servlet instance
   * Initialization setp up any resources JSP migh need such db

4. Requesting Processing Phase:
   * The servlet Process-incoming client request by calling service()
   * This phase is where dynamic content generation occurs

5. Destroy phase
   * The servlet container destroys servlet intance the destroy ()
   * This phase ensure resources are properly released.

rvlet by

converted int
execute.

tion phase
nto executable

stance
migh

st by

generation

intance

operly

3.

PHP code :

```
<! Doctype html>
<html lang. "en" >
<head>
  <meta charset = "UTF -8">
  <meta name = "view port"
content = " width : device -width, initial -scale
  <title> chessboard </title>
  <style>
    table {
      border : collapse: collapse;
      width: 400px;
      heigth : 400 px;
    }
  td {
      width : 30px;
      height :30px;
   for ($row =0; $row < 8; $row++){
        echo "<tr>";
     for ($col =0; $col <8; $col++){
          echo "</tr>";
     }
     </table>
     </body>
     </html>
```

output :

```
[w] [B] [w] [B] [w] [B] [w] [B]
[B] [w] [B] [w] [B] [w] [B] [w]
[w] [B] [w] [B] [w] [B] [w] [B]
[B] [w] [B] [w] [B] [w] [B] [w]
[w] [B] [w] [B] [w] [B] [w] [B]
[B] [w] [B] [w] [B] [w] [B] [w]
[w] [B] [w] [B] [w] [B] [w] [B]
[B] [w] [B] [w] [B] [w] [B] [w]
```

*w represent a white cell

*B represent a black cell

4.

PHP . code for application

```
<? Php
$text file path :'input.txt';
$xml file path = "output. xml;
$text content: file-get-contents ($textfile path);
$email pattern = (a-z A-z0-9 -%.+ ]
$phone pattern = '/[b] {10} \b/;
Preg-match all ($phone pattern, $text content, $phones);
$xml =new
Simple xml element ('<Data/>');
$emails Element= $xml → add Child ('email');
Simple xml element ('<Data/');
$phone element = $xml ('phone numbers');
echo - Data extracted' and saved to file successfully;
}>
```

Output:

```
Email : Support @ example. com
Phone number : 1234567890
```