**III BTech I Sem Regular Exam – January 2024**

**Web Technologies (WT) Key**

**PART -A**

1. Inline style sheet – style attribute is used for defining the style for particular element.

**Syntax:**

**<element name style=”property\_ name1:value1;”/>**

Example Snippet code

<p style="color:blue;font-size:46px;"> I LIKE WEB DESIGNING, </p>

2. CSS – CSS stands for **Cascading Style Sheets**

* CSS describes how HTML elements are to be displayed on screen, paper, or in other media

**a.Inline style sheets**

**<h1 style="color:blue;text-align:center;">This is a heading</h1>**

**b.Internal Style Sheets**

**<style>  
 h1 {  
   color: orange;  
 }**

**c.External Style Sheets - External stylesheets are stored in CSS files**

Link tag used is used for referring the External style sheets as follows.

**<link rel="stylesheet" href="mystyle.css">**

1. Date Object in JavaScript

new Date() creates a date object with the **current date and time**:

example html snippet code:

<p id="demo"></p>

<script>

const d = new Date();

document.getElementById("demo").innerHTML = d;

</script>

**One can specify the format in the Date creation:**

**const d = new Date("2022-03-25");**

4.

JavaScript is a versatile and widely used programming language primarily used for building web applications and enhancing web pages. Here are some of its main characteristics and features:

**a. High-level Language** b. **Interpreted Language c. Dynamic Typing**

**d. Weakly Typed e. Prototype-based f. Event-Driven g. Single-threaded**

1. **Cross-platform i. Support for Asynchronous Programming**
2. **DOM Manipulation**
3. **AJAX (Asynchronous JavaScript and XML l. Modularity**

5. DTD stands for Document Type Definition.

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

Two ways to define Internal DTD and external DTD

Example:

<!DOCTYPE Message  
[  
<!ELEMENT Message (to,from,heading,body)>  
<!ELEMENT to (#PCDATA)>  
<!ELEMENT from (#PCDATA)>  
<!ELEMENT heading (#PCDATA)>  
<!ELEMENT body (#PCDATA)>  
]>

.6. JDBC driver :

The JDBC Driver is a software component that enables a Java application to interact with a database.

Java provides/supports JDBC API enables java application to interact with database and execute sql queries. 4 Types of JDBC Drivers.

7.Listing the major tasks of Servlets:

Servlets are Java-based components that extend the capabilities of servers to generate dynamic content.

Handling HTTP requests, Processing User form data, Handling sessions, Generating Dynamic content, Database connectivity, Implementing authentication and authorization, error handling etc .

8. The **Servlet Life Cycle** is the entire process of its creation till the destruction.

[Servlet web container](https://www.javastudypoint.com/2018/09/introduction-to-servlet.html) maintains the servlet lifecycle.

Three methods are central to the **life cycle of a servlet.**

These are init(), service() and destroy().

They are implemented by every servlet and are invoked at a specific time by the server.

service() overrides the doGet() and doPost() methods.

9. Advantages of JSP over Servlets: Simplified syntax, Separations of Concerns, Automatic session Management, easy Integrated with html and follows MVC architecture, build -in tag libraries, rapid development.

10. Various attributes of page directive element in JSP

Syntax: <%@ page attribute = "value" %>

Various Attributes : contentType, errorPage, IsErrorPage, extends, import, info, isThreadSafe, language, session, isElIgnorned

Example : <%@page contentType = "text/html" import = "java.util.Date" %>

**PART -B**

11.a. To override the CSS properties of a class using another class, we can use the **!important** directive.

In CSS, !important means **“this is important”**, and the **property:value** pair that has this directive is always applied even if the other element has higher specificity.

**Syntax:**

element1 {

property-x: value\_y !important; /\* This will be applied. \*/

}

element2 {

property-x: value\_z; /\* This will not be applied. \*/

}

11.b  **Animated Table using HTML and CSS**

**Step 1: Create Structure of Table using HTML:** We will create a table structure using a table tag in HTML.

**Step 2: Decorating Table using CSS:** Now, we will apply **CSS** over the table which we have created earlier.

**Example CSS snippet code for animation:**

/\* Applying css properties to

table components \*/

table,

td,

tr {

**padding**: 12px;

**color**: wheat;

**background**: indigo;

**border**: 1px solid black;

**border-collapse**: collapse;

**font-size**: 20px;

**font-family**: 'Lucida Sans',

        'Lucida Sans Regular',

        'Lucida Grande',

        'Lucida Sans Unicode',

        Geneva, Verdana, sans-serif;

}

/\* Apply css properties to th \*/

th {

**color**: white;

**border**: 1px solid black;

**border-collapse**: collapse;

**background**: cadetblue;

}

/\* Apply hover effect to td \*/

**td:hover {**

**background: orangered;**

**}**

12.a. Web Page to illustrate Image overlay hover

The image overlay hovers effect is adding another effect or image over the base image. The CSS overlay effect can create by using the following:

* It will contain two divisions; one will be the overlay division which will contain the image that will show up when you hover over the image.
* The other will be the container that will hold both the overlay and image.

<!DOCTYPE html>

<html>

<head>

<style>

.container {

  position: relative;

  width: 50%;

}

.image {

  display: block;

  width: 100%;

  height: auto;

}

.overlay {

  position: absolute;

  bottom: 0;

  left: 0;

  right: 0;

  background-color: #008CBA;

  overflow: hidden;

  width: 100%;

  height: 0;

  transition: .5s ease;

}

.container:hover .overlay {

  height: 100%;

}

.text {

  white-space: nowrap;

  color: white;

  font-size: 20px;

  position: absolute;

  overflow: hidden;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  -ms-transform: translate(-50%, -50%);

}

</style>

</head>

<body>

<h2>Slide in Overlay from the Bottom</h2>

<div class="container">

  <img src="./images/flowers2.jpg" alt="Avatar" class="image">

  <div class="overlay">

    <div class="text">Hello World</div>

  </div>

</div>

</body>

</html>

Note: we can create different Image overlay effects on hovering, fade in text, overlay slide, overleaf zoom etc …. Marks can be awarded for alternate solutions/programs.

12.b.

Changing the position of a scrollbar using CSS can be done by creating a new class for the element, targeting the scrollbar and the thumb, and then using the "position" property to change the position of the scrollbar.

Example Snippet:

<head>

<style>

 .scrollable-div{

            height: 150px;

            width: 250px;

            overflow-y: auto;

            background-color:pink;

            margin:auto;

            padding:5px;

            transform: rotate(180deg);

         }

        .scrollable-div-inside {

            transform: rotate(-180deg);

        }

         .scrollable-div::-webkit-scrollbar {

            width: 5px; /\* Set the width of the scrollbar \*/

            background-color: #F5F5F5; /\* Set the background color of the scrollbar \*/

            position: absolute;

            right: 0;

         }

         .scrollable-div::-webkit-scrollbar-thumb {

            background-color: #08b51f; /\* Set the color of the thumb \*/

         }

         ::-webkit-scrollbar-track {

            background: red;

            border-radius: 5px;

         }

      </style>

   </head>

   <body>

      <h3>The scroll bar on the Left side of the div element</h3>

      <div class="scrollable-div">

         <div class="scrollable-div-inside">some text sufficient to show scrollbar </div>

      </div>

   </body>

</html>

13.a. JavaScript program to present selected value in dropdown list.

<html>

<head>

   <title>Program to dispaly the selected options in a dropdown list</title>

</head>

<body>

   <h2>Using the <i>selectedIndex</i> property.</h2>

   <form name="dropForm">

      <select name="dropSelect" onchange="showDropInfo()">

         <option value="P">Select:</option>

         <option value="A">Option 1</option>

         <option value="B">Option 2</option>

         <option value="C">Option 3</option>

      </select>

   </form>

   <p id="pF"></p>

   <p id="pS"></p>

   <p id="pT"></p>

   <script>

      function showDropInfo() {

         var sT = dropForm.dropSelect;

         var pF = document.getElementById('pF');

         var pS = document.getElementById('pS');

         var pT = document.getElementById('pT');

         pF.innerHTML = ('Selected option index: ' + sT.selectedIndex);

         pS.innerHTML = ('Selected value: ' + sT[sT.selectedIndex].value);

         pT.innerHTML = ('Selected text: ' + sT[sT.selectedIndex].text);

      }

   </script>

</body>

</html>

13.b. Javascript to remove spaces from a String

trim()- removes the trailing and leading spaces.

Example Snippet to demonstrate how javascript is used to remove the spaces from the string.

<p>trim() removes whitespace from both sides of a string:</p>

<pre><p id="demo1"></p></pre>

<pre><p id="demo2"></p></pre>

<script>

let text = " Hello World! ";

let result = text.trim();

document.getElementById("demo1").innerHTML = text;

document.getElementById("demo2").innerHTML = result;

</script>

Note: Marks can be awarded/given if regular expressions are used to remove the spaces.

14. a. Different types of Selectors of JQuery:

jQuery selectors allow you to select and manipulate HTML element(s).

## **The element Selector:** The jQuery element selector selects elements based on the element name.You can select all <p> elements on a page like this:

$("p")

## **The #id Selector:**

 $("#test")

**The .class Selector:** The jQuery .class selector finds elements with a specific class.

$(".test")

**Other selectors : Attribute seletors:**

**$(“[href]”) - Selects all elements with an href attribute**

**$(“\*” ) selects all elements**

14.b. Java Script program to remove duplicates from an array of objects.

There are various ways to remove duplicates from an array of objects.

These are the following ways:

* Using [**filter() Method**](https://www.geeksforgeeks.org/javascript-array-filter-method/)
* Using [**set() Method**](https://www.geeksforgeeks.org/sets-in-javascript/)
* Using [**reduce() Method**](https://www.geeksforgeeks.org/javascript-array-reduce-method/)
* Using **[indexOf() Method](https://www.geeksforgeeks.org/javascript-array-indexof-method/)**
* Using **[forEach() Method](https://www.geeksforgeeks.org/javascript-array-foreach-method/)**

Example using filter():

let arr = ["apple", "mango", "apple",

"orange", "mango", "mango"];

function removeDuplicates(arr) {

return arr.filter((item,

index) => arr.indexOf(item) === index);

}

console.log(removeDuplicates(arr));

### 15.a. A DTD is a Document Type Definition.

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

### Internal and external DTDs.

### If the DTD is declared internal inside the XML file, it must be wrapped inside the <!DOCTYPE> definition.

### <?xml version="1.0"?> <!DOCTYPE Message [ <!ELEMENT Message (to,from,heading,body)> <!ELEMENT to (#PCDATA)>

### ]> followed by XML data.

### If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:

### <?xml version="1.0"?> <!DOCTYPE note SYSTEM "note.dtd"> Followed by xml document …

### Note,dtd is the external dtd.

### \*Using Both Internal and External DTDs

One can use both internal and external DTDs at the same time, using these forms of **<!DOCTYPE>** element and **<!DOCTYPE rootname SYSTEM url>** in the same xml document

Example:

**<?xml version="1.0" ?>**

**<!DOCTYPE tutorials SYSTEM "tutorials.dtd" [**

**…. Internal dtd**

**] > XML document**

Internal DTD follows the external DTD declaration and then XML data. In the above example **tutorials.dtd** is the external dtd referenced in the current document

15. b. The JDBC Driver is a software component that enables a Java application to interact with a database.

Java provides/supports JDBC API enables java application to interact with database and execute sql queries.

Types of JDBC Drivers:

Type 1: JDBC-ODBC bridge driver

Type 2: JDBC Native-API driver

Type 3: JDBC-Net pure driver

Type 4: JDBC Thin driver or 100% Pure Java driver

16.a. differences between setMaxRows(int) and setFetchSize(int)

**setFetchSize(int)** defines the number of rows that will be read from the database when the ResultSet needs more rows. The method in the java.sql.Statement interface will set the ‘default’ value for all the ResultSet derived from that Statement;

**setMaxRows(int)** sets the limit of the maximum number of rows in a ResultSet object. If this limit is exceeded, the excess rows are “silently dropped”.

* Both are defined in java.sql.Statement interface
* public void setMaxRows(int max)

public void setFetchSize(int) --- default value ‘0’

16.b. return types of execute(), executeQuery(), executeUpdate()

**- execute(), executeQuery(), executeUpdate()** methods are provided/defined by all the three statement objects( Statement, PreparedStatement, CallableStatement)

**executeQuery():** This method is used to execute statements that returns tabular data (example select). It returns an object of the class ResultSet.

Example : PreparedStatement stmt = con.prepareStatement(“ select \* from employee where empid=?);

Stmt.setInt(1,24); // even other Statement objects could be used.

    ResultSet rs = stmt.executeQuery("Select \*from Employee");

**executeUpdate():** This method is used to execute statements such as insert, update, delete. It returns an integer value representing the number of rows affected.

Int rowseffected= stmt.executeUpdate();

**execute( )** – execute sql statement and return Boolean value

boolean b= stmt.execute();

17.a. There are two types of packages in Java Servlet that are providing various functioning features to servlet Applications. The two packages are as follows:

1. javax.servlet package
2. javax.servlet.http package

Major Classes of javax.servlet package

1. GenericServlet
2. ServletInputStream
3. ServletOutputStream
4. ServletRequestWrapper
5. ServletResponseWrapper
6. ServletRequestEvent
7. ServletContextEvent
8. ServletRequestAttributeEvent
9. ServletContextAttributeEvent
10. ServletException
11. UnavailableException

17.b. Major Interfaces of javax.servlet

1. Servlet
2. ServletRequest
3. ServletResponse
4. RequestDispatcher
5. ServletConfig
6. ServletContext
7. SingleThreadModel
8. Filter
9. FilterConfig
10. FilterChain
11. ServletRequestListener
12. ServletRequestAttributeListener
13. ServletContextListener
14. ServletContextAttributeListener

18.a .Processing of HTTP post request by Servlets

In html form: Servlets handle HTTP post request by overriding doPost(). In html form, method attribute need to have “post” value.

**<form action=”servleturl” method=”post” >**

**// Inside servlet class file:**

**public** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

//business/request processing logic

doGet(request, response); // doPost calling doGet()to handle request.

}

public void doGet()(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

//business/ request processing logic

}

18.b. Security issues in Servlets:

Servlets are susceptible to various attacks as follows:

a) **Session Management:** Improper session management in servlets can lead to session hijacking or fixation. Developers need to ensure secure session handling, use secure cookies, and avoid passing sensitive information in URLs.

b) **Cross-Site Scripting (XSS):** Servlets can be susceptible to XSS attacks if they render user inputs without proper validation or escaping. Input validation and output encoding are essential to mitigate this risk.

c) **Injection attacks:** Injection Attacks:\*\* Servlets might be vulnerable to various injection attacks, including SQL injection, where malicious SQL queries are injected into input fields. Developers should use prepared statements or parameterized queries to prevent these attacks.

d**) Authentication and Authorization:**Weak authentication and authorization mechanisms can lead to unauthorized access to resources. It’s crucial to implement strong authentication methods and proper access controls.

**Error Handling and Information Leakage:**

**Denial of Service (DoS) Attacks:**

**File Upload Security:**

Note: To mitigate these security risks in servlets, developers should follow best practices such as input validation, output encoding, implementing secure authentication mechanisms, using HTTPS, keeping software up-to-date, and regularly conducting security audits and testing.

19.a. Different Types of Scripting Elements in JSP:

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

* scriptlet tag

A scriptlet tag syntax is as follows:

**<% java source code %>**

* expression tag : evaluates java expressions/ function call…..

**Syntax: <%= expression %>**

* declaration tag

This Tag is used to declare the variables / methods.

**Syntax: <%! Initialize variables %>**

19.b <%

         Date date = new Date();

         out.print( "<h2 >”+date.toString()+"</h2>");

      %>

* JSP program for displaying current date and time., giving welcome message accepting user ‘s details.
* JSP application computing incometax of a User on accepting user’s salary details.
* JSP application accepting user details through Registration form and updating the database, displaying the user database on demand…..
* JSP application validating User login credentials
* JSP application accepting request parameters such as ‘age’ from user and checking whether the user is eligible for voting or not.
* Jsp standard action tags

20.a. Sharing data between JSP pages

JSP provides different *scopes* for sharing data objects between pages, requests, and users.

The scope defines how long the object is available and whether it's available only to one user or to all application users.

Example snippet code:

Step1 :

<%

ArrayList<Student> std = **new** ArrayList<Student>();

 std.add(**new** Student("Virat Kohli", 20, "CSE"));

request.setAttribute("data", std);

// Creating a RequestDispatcher object to dispatch the request the request to another resource

RequestDispatcher rd =   request.getRequestDispatcher("stdlist.jsp");

rd.forward(request, response);

%>

Stdlist.jsp //snippet code --- accessing the data from std.jsp

  <%ArrayList<**Student**> std =

            (ArrayList<**Student**>)request.getAttribute("data");

        for(Student s:std){%>

        <%-- Arranging data in tabular form

        --%>

            <**tr**>

                <**td**><%=s.getName()%></**td**>

                <**td**><%=s.getAge()%></**td**>

                <**td**><%=s.getCrs()%></**td**>

            </**tr**>

            <%}%>

The following scopes are defined: *page*, *request*, *session*, and *application*. Forward action tags could be used for sharing the data between JSP pages/requests/users. A diagram of a computer

Description automatically generated

20.b.Error Handling and debugging mechanism in JSP pages.

There are two ways of handling exceptions in JSP. They are:

* **By errorPage and isErrorPage attributes of page directive**
* **By <error-page> element in web.xml file**
  1. **Handling Exception using page directive attributes**
* **errorPage**: Used to site which page to be displayed when exception occurred.  
   **Syntax : <%@page errorPage="url of the error page"%>**
* **isErrorPage**: Used to mark a page as an error page where exceptions are displayed.   
  Syntax : **<%@page isErrorPage="true"%>**
  1. **Handling Exceptions Using error-page Element through web.xml File**

<web-app>

<error-page>

<exception-type>java.lang.Exception</exception-type>

<location>/error.jsp</location>

</error-page>

</web-app>