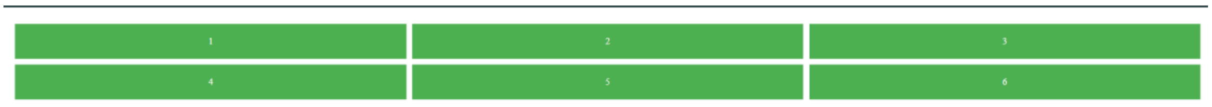
1. Create a web page using the advanced features of CSS Grid. Apply transitions and animations to the contents of the web page.

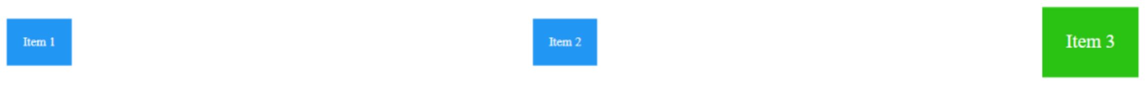
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Grid Page</title>
    <style>
      .grid-container {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
        gap: 10px;
        padding: 20px;
      }
      .grid-item {
        background-color: #4CAF50;
        color: white;
        padding: 20px;
        text-align: center;
        transition: transform 0.3s;
      }
      .grid-item:hover {
        transform: scale(1.1);
      }
    </style>
  </head>
  <body>
```

```
        <div class="grid-container">
            <div class="grid-item">1</div>
            <div class="grid-item">2</div>
            <div class="grid-item">3</div>
            <div class="grid-item">4</div>
            <div class="grid-item">5</div>
            <div class="grid-item">6</div>
        </div>
    </body>
</html>
```



2. Create a web page using the advanced features of CSS Flexbox. Apply transitions and animations to the contents of the web page.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>CSS Flexbox Page</title>
 <style>
 .flex-container {
 display: flex;
 justify-content: space-around;
 padding: 20px;
 }
 .flex-item {
 background-color: #2196F3;
 color: white;
 padding: 20px;
 text-align: center;
 transition: transform 0.29s;
 }
 .flex-item:hover {
   transform: scale(1.5) rotate(360deg);
 background-color: #2ac213;
 }
 </style>
</head>
<body>
 <div class="flex-container">
 <div class="flex-item">Item 1</div>
 <div class="flex-item">Item 2</div>
 <div class="flex-item">Item 3</div>
 </div>
</body>
</html>
```



3. Demonstrate pop-up box alerts, confirm, and prompt using JavaScript.

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>JavaScript Alerts</title>
</head>
<body>
 <button onclick="showAlert()">Show Alert</button>
 <button onclick="showConfirm()">Show Confirm</button>
 <button onclick="showPrompt()">Show Prompt</button>
 <script>
 function showAlert() {
 alert("This is an alert box!");
 }
 function showConfirm() {
 if (confirm("Are you sure?")) {
 alert("You pressed OK!");
 } else {
 alert("You pressed Cancel!");
 }
 }
 function showPrompt() {
 const name = prompt("Please enter your name:");
 if (name) {
 alert("Hello, " + name + "!");
 }
 }
 </script>
</body>
</html>
```

4. Demonstrate Responsive Web Design using Media Queries to create a webpage.

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Responsive Design</title>
 <style>
 body {
 font-family: Arial, sans-serif;
 }
 .container {
 width: 80%;
 margin: auto;
 background-color: lightgray;
 padding: 20px;
 }
 @media (min-width: 1000px) {
   body{
   background-color: green;
   }
   .container {
 width: 100%;
 }
 }
 @media (min-width: 600px) and (max-width: 1000px) {
   body{
    background-color: blue;
    }
    .container {
    width: 100%;
    }
    }
 </style>
</head>
<body>
 <div class="container">
 <h1>Responsive Web Page</h1>
 <p>This page adjusts based on screen size.</p>
 </div>
</body>
</html>
```

5. Write a JavaScript program to demonstrate the working of callbacks, promises, and async/await.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="r"></p>
    <script>
```

```
        function updateTime(callback)
        {
            setInterval(()=>{
                const date=new Date();
                callback(date);
            },1000);
        }
        function getTime()
        {
            return new Promise((resolve)=>{
                updateTime(resolve);
            });
        }
        async function showTime()
        {
            const date=await getTime();
            document.getElementById('r').innerHTML=date;
            showTime();
        }
        showTime();
    </script>
</body>
</html>
```

Separate

Callback

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function greet(name, callback) {
            console.log("Hello " + name);
            callback();
        }

        function goodbye() {
            console.log("Goodbye!");
        }

        // Calling greet function with goodbye as callback
        greet("Renu", goodbye);
    </script>
</body>
</html>
```

Promise

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
```

```
        function checkNumber(num) {
            return new Promise((resolve, reject) => {
                if (num > 10) {
                    resolve("Number is greater than 10!");
                } else {
                    reject("Number is 10 or less.");
                }
            });
        }

        checkNumber(15)
            .then((message) => console.log(message))  // If resolved
            .catch((error) => console.log(error));     // If rejected
    </script>
</body>
</html>
```

Async/Await

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        function delayMessage() {
            return new Promise((resolve) => {
                setTimeout(() => {
```

```
                resolve("This is an async message!");
            }, 3 );
        });
    }
    async function displayMessage() {
        const message = await delayMessage();
        console.log(message);
    }

    displayMessage();

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

6. Write an XML file that displays book information with the following fields: Title of the book, Author Name, ISBN number, Publisher name, Edition, and Price. Define a Document Type Definition (DTD) to validate the XML document created above.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Books [
  <!ELEMENT Books (books+)>
  <!ELEMENT books (title,author,ISBN,price)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT ISBN (#PCDATA)>
  <!ELEMENT price (#PCDATA)>
  ]>
```

```
<Books>
   <books>
      <title>ABC</title>
      <author>DEF</author>
      <ISBN>908</ISBN>
      <price>90</price>
   </books>
   <books>
      <title>ADE</title>
      <author>REF</author>
      <ISBN>897</ISBN>
      <price>90</price>
   </books>
</Books>
```

7. Write an XML file that displays book information with the following fields: Title of the book, Author Name, ISBN number, Publisher name, Edition, and Price. Define an XML schema to validate the XML document created above.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name="books">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="book" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="title" type="xs:string"/>
```

```
                <xs:element name="author" type="xs:string"/>
                <xs:element name="isbn" type="xs:string"/>
                <xs:element name="publisher" type="xs:string"/>
                <xs:element name="edition" type="xs:int"/>
                <xs:element name="price" type="xs:decimal"/>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
  </xs:element>
</xs:schema>
```

8. Write a Java application to validate the XML document using the DOM parser.

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

public class Eight {
  public static void main(String[] args) {
    try {
      System.out.println("Enter the name of the XML document:");
      BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
      String filename = input.readLine();
      File fp = new File(filename);
```

```
        if (fp.exists()) {
            try {
                DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = dbFactory.newDocumentBuilder();
                Document doc = builder.parse(fp);

                System.out.println(filename + " is well-formed");

                NodeList elements = doc.getElementsByTagName("*");
                System.out.println("Following are the elements in: " + filename);



                for (int i = 0; i < elements.getLength(); i++) {
                    Element element = (Element) elements.item(i);
                    System.out.println("\t" + element.getNodeName());
                }

            } catch (Exception e) {
                System.out.println(filename + " is not well-formed.");
                System.exit(1);
            }
        } else {
            System.out.println("File not found");
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

9. Write a Java application to validate the XML document using the SAX parser.

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;

public class nine {
    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter any xml file name: ");
            String filename = br.readLine();
            File f = new File(filename);

            if (f.exists()) {
                try {
                    SAXParserFactory db = SAXParserFactory.newInstance();
                    SAXParser builder = db.newSAXParser();
                    System.out.println("\nElements in the xml document are:");

                    DefaultHandler handler = new DefaultHandler() {
                        @Override
                        public void startElement(String uri, String localName, String qName,
Attributes attributes) throws SAXException {
                            System.out.println("\t" + qName);
                        }
                    };
                    builder.parse(f, handler);
                    System.out.println("\n" + filename + " is a well-formed XML Document");
                } catch (Exception e) {
                    System.out.println(e.getMessage());
                }
            } else {
                System.out.println("File not found");
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

10. Write a Java program to access the metadata of an SQL database.

```
import java.sql.*;
public class App {
    private static final String url = "jdbc:mysql://localhost:3306/cvr";
    private static final String user = "root";
    private static final String pwd = "Root@143#";
    public static void main(String[] args) throws Exception {
        Class.forName("com.mysql.cj.jdbc.Driver");
        System.out.println("Driver Connected...");
        Connection con = DriverManager.getConnection(url, user, pwd);
        Statement t=con.createStatement();
        String q = "Select * from student";
        ResultSet r = t.executeQuery(q);
        ResultSetMetaData rmd = r.getMetaData();
        int count = rmd.getColumnCount();
        System.out.println(rmd.getTableName(count));
        for(int i=1;i<=count;i++){
            System.out.println(rmd.getColumnName(i) + "\t\t\nColumn
Type:"+rmd.getColumnType(i));
        }
    }
}
```

11. Java script scientific Calculator

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Calculator</title>
  <style>
    * {
        box-sizing: border-box;
      }

      body {
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        font-family: Arial, sans-serif;
      }
```

```
.calculator {
  width: 200px;
  border: 2px solid #333;
  border-radius: 8px;
}

#display {
  width: 100%;
  height: 50px;
  font-size: 1.5em;
  text-align: right;
  padding: 10px;
  border: none;
  background-color: #f0f0f0;
}

.buttons {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 5px;
  padding: 5px;
}

button {
  font-size: 1.2em;
  padding: 10px;
  background-color: #ddd;
  border: none;
  cursor: pointer;
```

```
      }
      button:hover {
        background-color: #bbb;
      }
  </style>
</head>
<body>
  <div class="calculator">
    <input type="text" id="display" disabled>
    <div class="buttons">
      <button onclick="clearDisplay()">C</button>
      <button onclick="deleteLast()">←</button>
      <button onclick="appendOperator('/')">÷</button>
      <button onclick="appendOperator('*')">×</button>
      <button onclick="appendNumber('7')">7</button>
      <button onclick="appendNumber('8')">8</button>
      <button onclick="appendNumber('9')">9</button>
      <button onclick="appendOperator('-')">−</button>
      <button onclick="appendNumber('4')">4</button>
      <button onclick="appendNumber('5')">5</button>
      <button onclick="appendNumber('6')">6</button>
      <button onclick="appendOperator('+')">+</button>
      <button onclick="appendNumber('1')">1</button>
      <button onclick="appendNumber('2')">2</button>
      <button onclick="appendNumber('3')">3</button>
      <button onclick="calculate()">=</button>
      <button onclick="appendNumber('0')">0</button>
      <button onclick="appendNumber('.')">.</button>
      <button></button>
      <button></button>
    </div>
  </div>
  <script>
    function appendNumber(number) {
      document.getElementById("display").value += number;
    }

    function appendOperator(operator) {
      document.getElementById("display").value += operator;
    }

    function clearDisplay() {
      document.getElementById("display").value = "";
    }

    function deleteLast() {
      let display = document.getElementById("display");
      display.value = display.value.slice(0, -1);
    }

    function calculate() {
      let display = document.getElementById("display");
      try {
        display.value = eval(display.value);
      } catch (error) {
        display.value = "Error";
      }
    }
  </script>
</body>
```

```
</html>
```

12. Demonstrate Servlet Lifecyle by implementing Servlet Interface

```
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

public class SampleServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

    public SampleServlet() {
        super();
    }
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		response.setContentType("text/html");
		PrintWriter pw = response.getWriter();
		pw.println("<html><head><title>My First Sample Servlet</title></head>");
		pw.println("<body>");
		pw.println("<h1>Hello</h1>");
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		doGet(request, response);
	}

}
```

```
<servlet>
    <servlet-name>SampleServlet</servlet-name>
    <servlet-class>SampleServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SampleServlet</servlet-name>
    <url-pattern>/servlets/servlet/SampleServlet</url-pattern>
  </servlet-mapping>
```

13. Demonstrate Creation of Servlet program using Http Servlet class

```
import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
```

```
public class ServletLifeCycle extends HttpServlet {
        private static final long serialVersionUID = 1L;
   ServletConfig config;
   public ServletLifeCycle() {
      super();
   }

        public void init(ServletConfig config) throws ServletException {
                this.config = config;
        }

        public void destroy() {
                config=null;
        }

        protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                response.setContentType("text/html");
                PrintWriter pw = response.getWriter();
                pw.println("<h1>Hello</h1>");
        }
}
```