Rustamji Institute of Technology

BSF Academy, Tekanpur, Gwalior M.P.

Department of Computer Science & Engineering

5th Semester

Subject: Internet & Web Technology

(CS - 504)

Theory Assignment No. 4

Assignment Details

Topic: XML, DTD, DOM, and PHP Basics

Submission Deadline: 15th December 2024

Format: Theory & Practical Code

Assignment Solutions

Q1. What is XML? What are the attributes of an XML?

1.1 Definition of XML

XML stands for **Extensible Markup Language**. It is a text-based markup language derived from Standard Generalized Markup Language (SGML). Unlike HTML, which is designed to display data (focusing on looks), XML is designed to store and transport data (focusing on structure and content).

Key Characteristics:

- **Extensible:** XML tags are not predefined. You must define your own tags (e.g., `<student>`, `<salary>`).
- **Platform Independent:** XML data is stored as plain text, making it software- and hardware-independent.
- **Self-descriptive:** An XML document carries information about its own structure.
- **Strict Syntax:** Unlike HTML, XML is case-sensitive and requires all tags to be properly closed and nested.

1.2 Attributes in XML

XML attributes provide additional information about elements. Attributes are designed to contain data related to a specific element but not part of the data content itself.

Rules for Attributes:

(a) Attributes must appear in the **start tag**.

(b) Attribute values must always be enclosed in **quotes** (single or double).

(c) An element cannot have multiple attributes with the same name.

(d) Attributes generally store metadata (data about data), such as IDs or types.

Example:

```
<university>
    <student id="CS101" category="undergrad">
        <name>Rahul Sharma</name>
        <branch>CSE</branch>
    </student>
</university>
```

Q2. What is DTD? Why do we use it?

2.1 Definition

DTD stands for **Document Type Definition**. It is a set of rules that defines the structure and the legal elements and attributes of an XML document. A DTD can be declared inline inside the XML document or as an external reference.

2.2 Purpose and Usage

We use DTDs for the following critical reasons:

- **Validation:** DTDs allow XML parsers to verify that an XML document is "Valid". If the XML does not follow the DTD rules (e.g., missing a required tag), the parser will throw an error.
- **Standardization:** Large organizations use DTDs to agree on a standard format for data exchange. For example, two banks exchanging transaction data will use a shared DTD to ensure the data format matches.
- **Documentation:** A DTD serves as documentation for the XML file, explaining what elements are allowed, their order, and their hierarchy.

Example Declaration: `<!DOCTYPE note SYSTEM "note.dtd">`

Q3. Discuss the difference between HTML and XHTML with respect to elements?

XHTML (Extensible HyperText Markup Language) is a stricter, cleaner version of HTML. It is HTML defined as an XML application. The differences regarding elements are strict to ensure compatibility with XML parsers.

codeGray **HTML Elements**	**XHTML Elements**
Case Sensitivity: HTML tags are case-insensitive. `<BODY>` and `<body>` are the same.	**Case Sensitivity:** XHTML tags **must** be in lower case. `<BODY>` is invalid.
Closing Tags: Some elements like `<p>`, `<li>`, and `<td>` do not require closing tags.	**Closing Tags:** All non-empty elements **must** have a closing tag (e.g., `</p>`).
Empty Elements: Empty elements like break can be written as ` `.	**Empty Elements:** Must be self-closed (e.g., ` ` or `<hr />`).
Nesting: Improper nesting is often tolerated (e.g., `<b><i>text</b></i>`).	**Nesting:** Elements must be correctly nested (e.g., `<b><i>text</i></b>`).
Root Element: No strict requirement for a single root in fragments.	**Root Element:** The document must have exactly one root element (`<html>`).

Q4. How can the XML document be transformed into HTML document using XSLT? Write a suitable code for the same.

4.1 XSLT Overview

XSLT (Extensible Stylesheet Language Transformations) is the recommended style sheet language for XML. While CSS is used to style HTML, XSLT is used to *transform* XML.

It works by matching templates to elements in the source XML tree and creating a new result tree (usually HTML).

4.2 Source XML (class.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
<class>
  <student>
    <name>Amit Kumar</name>
    <marks>85</marks>
  </student>
  <student>
    <name>Priya Singh</name>
```

```
    <marks>92</marks>
  </student>
</class>
```

4.3 XSLT Code (style.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>Student Results</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Name</th>
        <th>Marks</th>
      </tr>
      <xsl:for-each select="class/student">
      <tr>
        <td><xsl:value-of select="name"/></td>
        <td><xsl:value-of select="marks"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Q5. How is the XML document parsed into HTML document using JavaScript?

To manipulate XML data inside a web page, we must parse the XML string into a JavaScript object known as the **XML DOM** (Document Object Model).

The standard interface for this is the `DOMParser` object.

Steps involved:

(a) Create an instance of `new DOMParser()`.

(b) Use the `parseFromString()` method, passing the XML text and the content type `"text/xml"`.

(c) The result is an XML DOM object which can be traversed using standard DOM methods like `getElementsByTagName()` or `childNodes`.

Implementation Code:

```
<!DOCTYPE html>
<html>
<body>
    <h3>XML Parsing Example</h3>
    <p id="demo"></p>

    <script>
    var text = "<bookstore><book>" +
               "<title>Everyday Italian</title>" +
```

```
                "<author>Giada De Laurentiis</author>" +
                "</book></bookstore>";

    var parser = new DOMParser();
    var xmlDoc = parser.parseFromString(text, "text/xml");

    // Extracting the title
    var title = xmlDoc.getElementsByTagName("title")[0].childNodes
    [0].nodeValue;

    document.getElementById("demo").innerHTML = "Book Title: " +
    title;
    </script>
</body>
</html>
```

Q6. Explain the different components of DTD with example?

A DTD defines the validity of an XML document using four main components:

(a) **Elements:** Elements are the main building blocks of both XML and HTML files. In a DTD, elements are declared with an ELEMENT declaration.
Syntax: `<!ELEMENT element-name (content-model)>`
Example: `<!ELEMENT note (to, from, heading, body)>`

(b) **Attributes:** Attributes provide extra info about elements. They are declared with an ATTLIST declaration.
Syntax: `<!ATTLIST element-name attribute-name attribute-type default-value>`
Example: `<!ATTLIST payment type CDATA "check">`

(c) **Entities:** Entities are variables used to define shortcuts to common text. Entity references are references to entities.
Syntax: `<!ENTITY entity-name "entity-value">`
Example: `<!ENTITY writer "Donald Duck">` (Usage: `&writer;`)

(d) **PCDATA (Parsed Character Data):** PCDATA is text that will be parsed by a parser. Tags inside the text will be treated as markup and tags will be expanded.

(e) **CDATA (Character Data):** CDATA is text that will *not* be parsed by a parser. This is used for text blocks containing characters that would otherwise be interpreted as markup (like < and >).

Q7. Explain in detail DOM event handling. Also explain with an example of creating a context menu?

7.1 DOM Event Handling Phases

When an event (like a click) occurs on a DOM element, it doesn't just affect that element. It travels through the DOM tree. The standard DOM event flow has three phases:

(a) **Capturing Phase:** The event starts from the root (`html`) and travels down the tree to the target element.

(b) **Target Phase:** The event reaches the target element (the element that was clicked).

(c) **Bubbling Phase:** The event bubbles up from the target element back to the root.

Developers can choose to handle events during the capturing or bubbling phase using `addEventListener(event, function, useCapture)`.

7.2 Creating a Custom Context Menu

A context menu is the menu that appears on a right-click. By default, the browser shows its own menu. We can override this using the `contextmenu` event.

```
<!DOCTYPE html>
<html>
<style>
  #myMenu {
      display: none;
      position: absolute;
      background-color: #f9f9f9;
      box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
      padding: 12px;
      border: 1px solid #ddd;
  }
</style>
<body>

<div id="myMenu">
  <p>Action 1: Copy</p>
  <p>Action 2: Paste</p>
</div>

<script>
document.addEventListener('contextmenu', function(e) {
    e.preventDefault(); // Important: Stop the default browser menu

    var menu = document.getElementById("myMenu");
    menu.style.display = 'block';

    // Position the menu where the mouse clicked
    menu.style.left = e.pageX + "px";
    menu.style.top = e.pageY + "px";
});

// Hide menu when clicking elsewhere
document.addEventListener('click', function() {
    document.getElementById("myMenu").style.display = "none";
});
</script>

</body>
</html>
```

Q8. Explain the features of Date and Time format in PHP?

PHP provides a powerful function `date(format, timestamp)` to handle date and time. If the timestamp is omitted, the current date/time is used.

Common Format Characters:

- **d**: Day of the month, 2 digits with leading zeros (01 to 31).
- **D**: A textual representation of a day, three letters (Mon to Sun).
- **m**: Numeric representation of a month, with leading zeros (01 to 12).

- **M**: A short textual representation of a month, three letters (Jan to Dec).
- **Y**: A full numeric representation of a year, 4 digits (e.g., 2024).
- **H**: 24-hour format of an hour (00 to 23).
- **i**: Minutes with leading zeros (00 to 59).
- **s**: Seconds with leading zeros (00 to 59).
- **a**: Lowercase Ante meridiem and Post meridiem (am or pm).

Example Usage:

```php
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "The time is " . date("h:i:s a");
?>
```

Q9. What is the difference between GET and POST method in PHP?

codeGray **Feature**	**GET Method**	**POST Method**
Purpose	Used to request data from a specified resource.	Used to send data to a server to create/update a resource.
Visibility	Data is sent in the URL (Query String). Everyone can see it.	Data is sent in the HTTP Message Body. It is hidden from the user.
Security	**Low**. Never use GET for sensitive data like passwords.	**High**. Suitable for sensitive data.
Data Limit	Limited (approx 2048 characters) because URLs have length limits.	No specific limit. Can send large files/images.
Bookmarking	Results can be bookmarked and cached.	Cannot be bookmarked or cached.
Data Type	Only ASCII characters allowed.	Binary data is also allowed (e.g., file uploads).

Q10. What is Environment variable in PHP and what is the use of it?

Definition: Environment variables are special global variables that allow the script to interact with the environment in which it is running (the web server, the browser, and the OS). In PHP, these are automatically populated in the **$_SERVER** superglobal array.

Common Uses and Variables:

- **Getting User IP:** `$_SERVER['REMOTE_ADDR']` gives the IP address of the user viewing the page.
- **Script Location:** `$_SERVER['PHP_SELF']` returns the filename of the currently executing script.
- **Server Details:** `$_SERVER['SERVER_NAME']` returns the name of the server host.
- **Request Method:** `$_SERVER['REQUEST_METHOD']` tells you if the page was accessed via GET, POST, etc.

These are crucial for security (checking referrers), logging (tracking IPs), and navigation (handling self-submitting forms).

Q11. Explain the concept of Object Oriented Programming with PHP?

PHP 5+ introduced a full object model. Object-Oriented Programming (OOP) allows developers to create modular, reusable code.

Key Concepts:

(a) **Class:** A template or blueprint for creating objects. It defines properties (variables) and methods (functions).

(b) **Object:** An instance of a class. It has the structure defined by the class but holds specific data.

(c) **Encapsulation:** Grouping data and methods together and restricting access using modifiers:

- `public`: Accessible from everywhere.
- `private`: Accessible only within the class.
- `protected`: Accessible within the class and derived classes.

(d) **Inheritance:** A child class can inherit properties and methods from a parent class using the `extends` keyword.

Example Code:

```php
<?php
class Car {
  public $model;

  // Constructor
  function __construct($model) {
    $this->model = $model;
  }

  function getModel() {
    return "The car is a " . $this->model;
  }
}

// Inheritance
class SportsCar extends Car {
    public function turbo() {
        return "Zoom!";
    }
}

$myCar = new SportsCar("Ferrari");
echo $myCar->getModel(); // Output: The car is a Ferrari
?>
```

Q12. What are the different modes of opening the file in PHP?

File handling in PHP is done using the `fopen(filename, mode)` function. The **mode** parameter specifies the type of access you require to the stream.

`'r'` **(Read Only):** Open for reading only. The file pointer starts at the beginning of the file.

`'r+'` **(Read/Write):** Open for reading and writing. The file pointer starts at the beginning.

`'w'` **(Write Only):** Open for writing only. Clears the file content (truncates) or creates a new file if it doesn't exist.

`'w+'` **(Read/Write):** Open for reading and writing. Clears the file content or creates a new file.

`'a'` **(Append):** Open for writing only. The data is written at the end of the file. Creates a new file if it doesn't exist.

`'a+'` **(Read/Append):** Open for reading and writing. Data is preserved and new data is added to the end.

`'x'` **(Create):** Creates a new file for writing only. Returns FALSE and an error if the file already exists.