IWP-TEE paper solution

Q1. Explain about following the protocol:

a) Web Server, b) Email, c) Static Web page, d) Dynamic web page.

Ans.

a) Web servers use the Hypertext Transfer Protocol (HTTP) to respond to requests from client browsers and deliver web pages and other content. HTTP is an application layer protocol that defines how messages are formatted and transmitted between servers and clients.

b) Email uses multiple protocols to send and receive messages:

- Simple Mail Transfer Protocol (SMTP) is used to send email from a mail client to an email server.
- Post Office Protocol (POP3) and Internet Message Access Protocol (IMAP) are used to retrieve email from a server.
- Multipurpose Internet Mail Extensions (MIME) encodes attachments and non-ASCII text in email.,

c) Static web pages have fixed, pre-written content that is served as simple HTML files. They display the same information to all users and do not change dynamically.

d) Dynamic web pages are generated on-the-fly when a user requests a page. They may pull information from databases and can provide customised content for each user. Common technologies for dynamic pages include server-side scripts (PHP, ASP.NET) and client-side JavaScript.

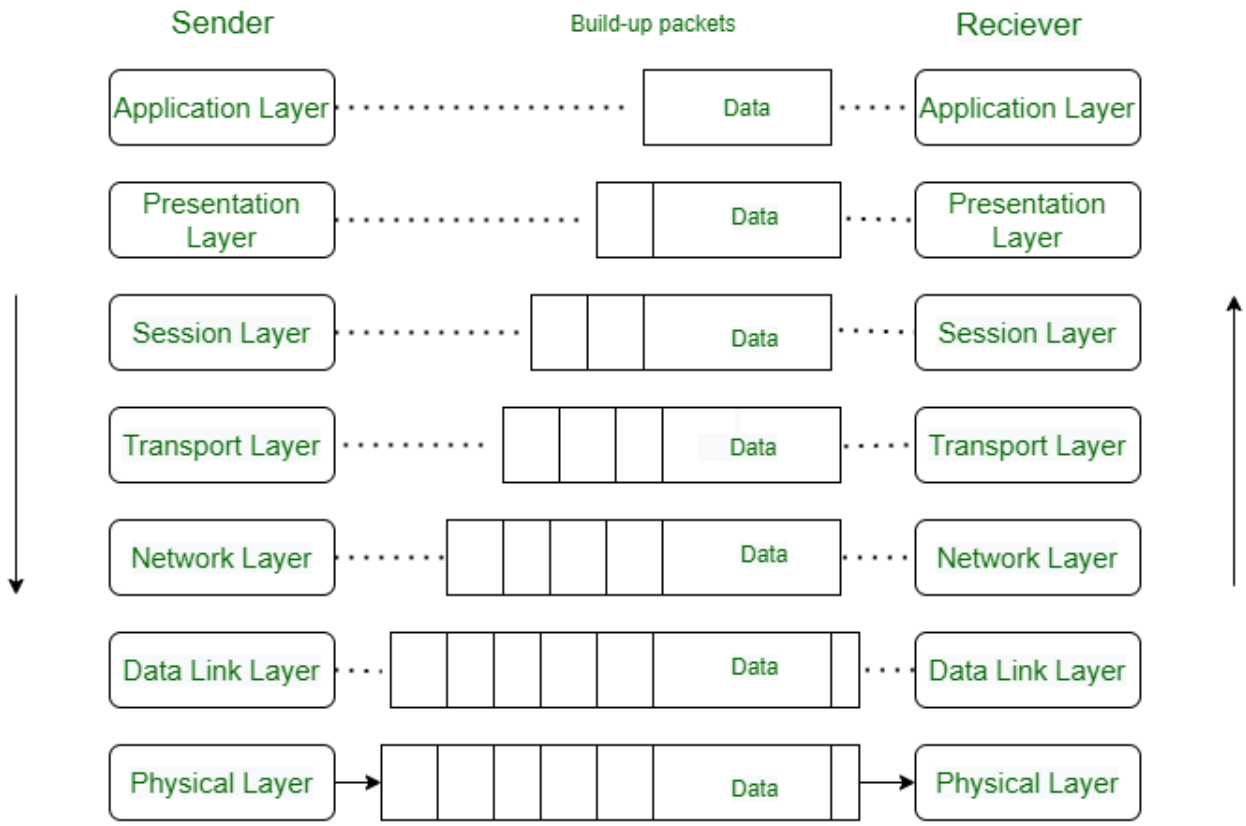
The key difference is static pages have pre-built content, while dynamic pages generate content at runtime based on variables like user input. Web servers and email follow common protocols, while static and dynamic web content differ in how they are created and delivered.

(b) Compare TCP/IP and OSI Models of networking with suitable diagram.

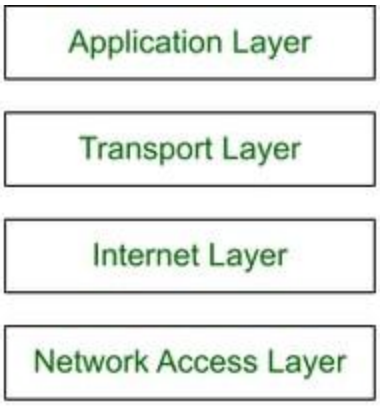
Ans.

Difference between OSI Model and TCP/IP Model

Parameters	OSI Model	TCP/IP Model
Full Form	OSI stands for Open Systems Interconnection.	TCP/IP stands for Transmission Control Protocol/Internet Protocol.
Layers	It has 7 layers.	It has 4 layers.
Usage	It is low in usage.	It is mostly used.
Approach	It is vertically approached.	It is horizontally approached.
Delivery	Delivery of the package is guaranteed in OSI Model.	Delivery of the package is not guaranteed in TCP/IP Model.
Replacement	Replacement of tools and changes can easily be done in this model.	Replacing the tools is not easy as it is in OSI Model.
Reliability	It is less reliable than TCP/IP Model.	It is more reliable than OSI Model.



OSI model



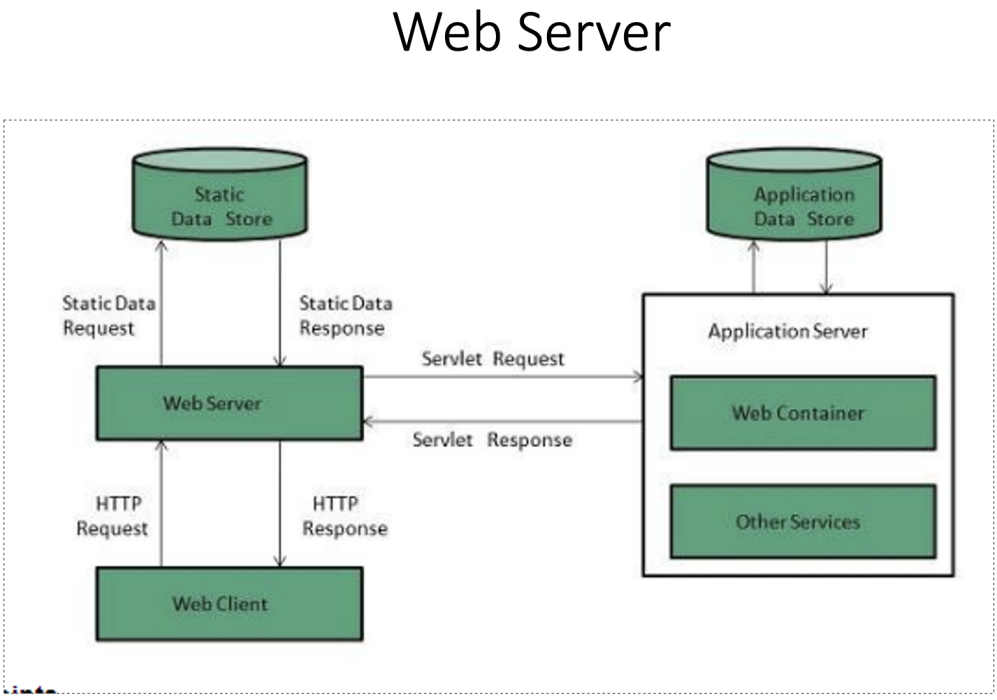
Various Layers of the TCP/ IP Model

TCP/IP model

OR

(c) Explain the concept of web server architecture and its working with a suitable diagram. Explain server side scripting technologies for creating acting web pages.

Ans.



Web Server Architecture:

1. A web server is a software or hardware system that serves websites and web applications to users over the internet.
2. It follows a client-server model, where the client (typically a web browser) sends requests to the server, and the server responds with the requested content.
3. The web server architecture consists of multiple components, including:
    - Client: The user's device that sends requests to the server.

- Web Browser: Software on the client's device that interacts with the server.
- Internet: The network infrastructure that connects the client and server.
- Web Server: The software or hardware system that receives and processes requests, generates responses, and sends them back to the client.
- Application Server: Handles server-side application logic and interacts with databases or other resources.
- Database Server: Stores and retrieves data requested by the application server.
- DNS Server: Translates domain names into IP addresses to locate the appropriate web server.

4. Working of Web Server Architecture:
    - The client initiates a request by entering a URL or clicking on a hyperlink in the web browser.
    - The web browser sends the request to the web server using the HTTP or HTTPS protocol.
    - The web server receives the request and identifies the requested resource (HTML page, image, file, etc.).
    - If the resource is static (e.g., HTML, CSS, images), the web server retrieves it from the file system and sends it back to the client.
    - If the resource requires dynamic content (e.g., generated HTML, database queries), the web server passes the request to the application server.
    - The application server processes the request, generates the required content, and sends it back to the web server.
    - The web server then sends the response to the client, which displays the content in the web browser.
    - This communication happens through a series of HTTP requests and responses.
    - The process continues as the client interacts with the website, sending new requests for additional resources.

Server-side scripting allows dynamic web pages by executing code on the server before sending HTML to the browser. Popular technologies are:

PHP - Open source, embedded within HTML and widely used. Interacts with databases.

ASP.NET - Uses C#/VB with .NET libraries. Integrates well with Microsoft technologies.

JSP - Uses Java and compiles into servlets. Can leverage full Java capabilities.

Ruby on Rails - Uses Ruby language and MVC architecture for rapid development.

These technologies process form data, access databases, control user access and integrate with other services on the server before delivering HTML to the client. They generate interactive web pages and enable rapid web application development.

Q 2. (a) In your views is CSS box model is essential or not? What do you understand by CSS box model, explain with help of suitable example? Explain different types of selectors in CSS with code.

Ans.

Yes, the CSS box model is an essential concept for web development as it allows us to layout and align page elements precisely.

The CSS box model represents every element on a page as a rectangular box. It consists of:

- Content - The actual content of the element, such as text, image etc.
- Padding - Space between the content and the border.
- Border - The edge around the content and padding.
- Margin - Space between the border and outside elements.

For example:

```
.box {
  width: 300px;
  padding: 20px;
```

```
  border: 5px solid gray;
  margin: 20px;
}
```

This will apply 300px width, 20px padding, 5px gray border and 20px margin to the element with class "box".

The main CSS selectors are:

- Element selector - Selects HTML elements, like `p { }`
- ID selector - Selects element with specific ID, like `#main { }`
- Class selector - Selects elements with specific class, like `.box { }`
- Universal selector - Selects all elements, like `{ }`
- Attribute selector - Selects elements with specific attributes, like `a[target] { }`
- Pseudo-class selector - Selects element state, like `a:hover { }`

So the box model and selectors allow targeting specific page elements for styling. They are foundational aspects of CSS.

(b) How events are triggered by actions inside a HTML form?

Ans.

HTML forms allow user interaction through various elements like text fields, buttons, checkboxes etc. These elements can trigger JavaScript events when a user performs actions on them. Some common ways events are triggered in a form are:

- onchange - Triggers when value of an input element like text, checkbox, select etc. changes.

```
<input type="text" onchange="validate(this)">
```

- onclick - Triggers when an element like button, checkbox, image is clicked.

```
<button onclick="submitForm()">Submit</button>
```

- onsubmit - Triggers when a form is submitted.<form onsubmit="validateForm()">

```
<form onsubmit="validateForm()">
```

- onfocus - Triggers when an input element gains focus.

```
<input type="text" onfocus="highlight(this)">
```

- onblur - Triggers when input element loses focus.

```
<input type="text" onblur="validateInput(this)">
```

- onselect - Triggers when text in input or textarea is selected.

So in summary, HTML form elements have built-in events that get triggered on user actions like change, click, submit etc. We can assign JavaScript event handler functions to these events to perform validation, data processing etc.

OR

(c) How are following form element are useful? Write code for implementation of following tags:

i) <input>

ii) <label>

iii) <select>

iv) <textarea>

v) <button>

vi) <fieldset>

Ans.

i) <input> - Allows different types of data entry:

```
<input type="text" name="username">
<input type="checkbox" name="terms">
```

ii) <label> - Labels an input element:

```
<label for="username">Username:</label>
<input id="username" type="text">
```

iii) <select> - Creates a dropdown list:

```
<select name="city">
  <option value="nyc">New York</option>
  <option value="la">Los Angeles</option>
</select>
```

iv) <textarea> - Multi-line text input:

```
<textarea name="message" rows="5" cols="30"></textarea>
```

v) <button> - Defines a clickable button:

```
<button type="submit">Submit</button>
```

vi) <fieldset> - Groups related elements:

```
<fieldset>
  <legend>Contact</legend>
  <label>Email:</label>
  <input type="email">
</fieldset>
```

These elements allow robust data entry, validation and submission in forms. The tags provide semantic structure and accessibility.

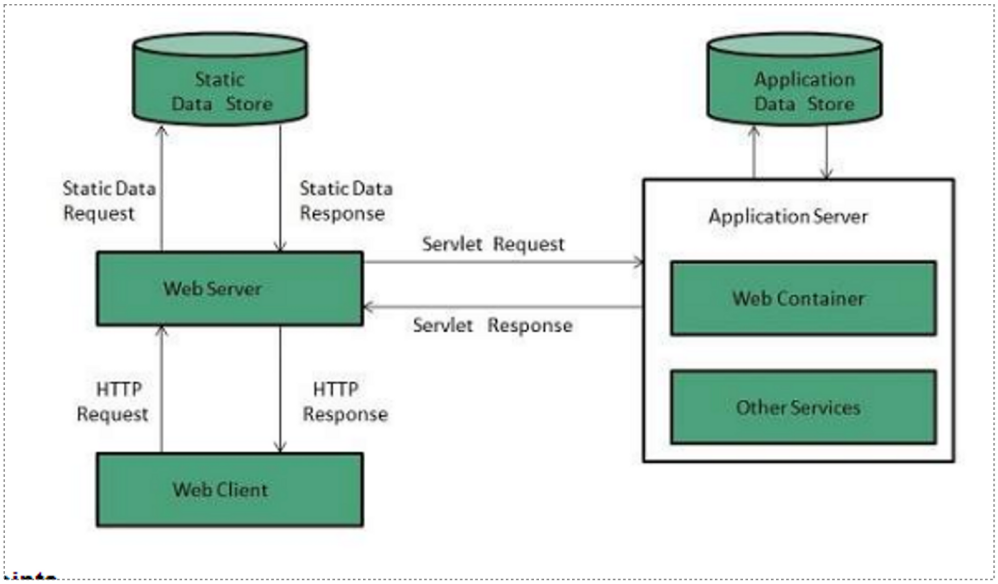
Q 3. (a) “JavaScript is an important client-side scripting language and widely used in dynamic websites”. Justify the statement in respect to client and server side scripting and also explain the essential web protocols in web server architecture with suitable diagram.

Ans.

You're right, JavaScript is an very important client-side scripting language and is widely used to create dynamic websites:

- Client-side Scripting: JavaScript code is executed on the client-side, i.e. the web browser. This allows interacting with the Document Object Model (DOM) to dynamically modify page content and react to user events.
- Server-side Scripting: Languages like PHP, Python, Ruby execute on the web server before the page is sent to the browser. Used for backend logic.
- JavaScript is client-side only, so ideal for frontend interactivity, validation, AJAX etc. without needing page reloads.
- JavaScript works across all major browsers and platforms. It can enhance UI, validate forms, call APIs etc.
- Almost all modern dynamic websites use JavaScript to provide responsive and seamless user experience.

Web Server



Web-Server Architecture

Some essential web protocols in a web server architecture are:

HTTP - Client-Server protocol for requesting and serving web pages/assets over the internet. Defines methods like GET, POST.

DNS - Resolves human friendly domain names to machine friendly IP addresses required for routing.

TLS/SSL - Provides encrypted communication between clients and servers for security.

TCP - Enables reliable transportation and sequencing of data packets between clients and servers.

IP - Defines addressing scheme to uniquely identify devices on the internet and route traffic.

These standard protocols allow clients like web browsers to communicate with remote servers to access websites and web applications securely over the internet. They enable reliable request-response messaging patterns and data transfer between distributed systems.

OR

(b) According to your views why validation and event handling is needed in JavaScript, explain it with the help of code. What is the use of document object model in JavaScript?

Ans.

Validation and event handling are important in JavaScript for the following reasons:

Validation:

- To ensure form data entered by the user is correct before submission.
- Avoid wrong data types, formats, empty values etc.
- Improves data quality and reduces errors.

For example:

```
function validateForm() {
  var name = document.getElementById("name").value;
  if (name == "") {
    alert("Name cannot be empty");
    return false;
  }
}
```

Event Handling:

- To detect actions performed by user like clicks, key presses etc.
- Execute relevant code when events occur.
- Allows interacting with users and dynamically updating UI.
- Provides rich user experience.

Example:

```
document.getElementById("submit").onclick = function() {
// code to submit form
}
```

The Document Object Model (DOM) in JavaScript is used to:

- Represent HTML documents as a structured object that can be manipulated.
- Dynamically access, add, modify page content, style and structure.
- Update UI, animations, user interaction without page reload.
- Improves performance and responsiveness of web apps.

So validation, events and DOM together allow interactive and dynamic web applications using JavaScript.

Q 4. (a) Write a procedure to find sum of two numbers in PL/SQL?

Ans.

A simple PL/SQL procedure to find the sum of two numbers:

```
CREATE OR REPLACE PROCEDURE find_sum
(
  num1 IN NUMBER,
  num2 IN NUMBER,
  sum_out OUT NUMBER
)
IS
BEGIN
  sum_out := num1 + num2;
END find_sum;
```

Explanation:

- CREATE OR REPLACE PROCEDURE defines a new procedure with the name find_sum.

- The procedure accepts two input parameters num1 and num2 of NUMBER datatype.
- An output parameter sum_out is declared to return the sum.
- The business logic simply adds num1 and num2 and assigns the result to sum_out.
- This sum_out can be used by the calling code to retrieve the sum.

To call this procedure:

```
DECLARE
  v_sum NUMBER;
BEGIN
  find_sum(10, 20, v_sum);
  DBMS_OUTPUT.PUT_LINE('Sum is ' || v_sum);
END;
```

So in PL/SQL, procedures allow encapsulating reusable business logic that can be invoked to perform specific tasks.

(b) What are different types of features in PL/SQL?

Ans.

PL/SQL provides various features to support procedural programming in Oracle database. The main features are:

1. Block Structure:

- PL/SQL code is written in blocks of code enclosed between BEGIN and END.
- This includes declarative, executable and exception handling sections.

2. Variables & Data Types:

- Supports variables and constants of types like NUMBER, VARCHAR2, BOOLEAN etc.

3. Control Structures:

- Supports conditional (IF/ELSE), looping (FOR LOOP, WHILE LOOP) and branching (GOTO) constructs.

4. Cursors & Exceptions:

- Cursor allows querying and processing result sets.
- Exceptions handle runtime errors gracefully.

5. Modular Programming:

- Code can be organised into procedures, functions, packages for reusability.

6. Object-Oriented Features:

- Supports object-oriented concepts like classes, objects, inheritance etc.

OR

(c) Explain about following the protocol: 1-HTTP 2-FTP

Ans.

HTTP (Hypertext Transfer Protocol) is an application layer protocol used for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web. A client such as a web browser sends an HTTP request message to a server, which returns a response message, typically containing the resource that was requested. HTTP is considered a stateless protocol as each request-response pair is executed independently.

FTP (File Transfer Protocol) is an application layer protocol used for transferring files between a client and server over a TCP connection. FTP enables uploading and downloading of files as well as traversing directories on the server. Unlike HTTP, FTP establishes separate connections for control commands and for transferring data. It is a stateful protocol that maintains the context of transfers across multiple requests. An FTP client logs into the server with a username and password before sending commands like GET, PUT, LIST, etc. The server authenticates the session and fulfills valid requests to access and transfer files.

Key differences between HTTP and FTP include:

- HTTP is primarily used for hypertext documents while FTP is designed for file transfers
- HTTP is stateless, FTP is stateful
- HTTP establishes one TCP connection, FTP uses separate connections for control and data transfer
- HTTP uses port 80 by default, FTP uses ports 20 and 21
- HTTP transactions are short-lived, FTP requires maintaining a session

In summary, HTTP and FTP are both application layer protocols fundamental to data communication, with HTTP being focused on hypertext transfer and FTP optimized for reliable file transfers. Following their standard specifications allows interoperable communication.

Q 5. (a) “PHP helps in making the dynamic pages”. Explain the statement in your own words. Develop and design the hospital registration form and store the data with the help of PHP and My-Sql database connectivity.

Ans.

"PHP helps in making dynamic pages" in my own words:

PHP is a server-side scripting language that can be used to generate dynamic web pages. By dynamic, it means the web pages are generated on-the-fly by executing PHP code each time the page is requested rather than serving static HTML pages. Some key ways PHP helps make pages dynamic are:

- PHP can connect to databases like MySQL to pull in data and display it on a page. So the content of the page can change based on the data.
- PHP allows you to store variables and execute logic like loops and conditionals when generating a page. So the page structure and flow can be dynamic.
- PHP can be used to build forms that accept user input and then process and respond to that input. This creates interactive, dynamic experiences.

- Cookies and sessions can be used in PHP to remember state and customize pages for individual users.

Here's a step-by-step breakdown of creating a hospital registration form using PHP and MySQL, following the approach you've described:

Step 1: Design the HTML Form (registration_form.php)

```
<!DOCTYPE html>
<html>
<head>
    <title>Hospital Registration Form</title>
</head>
<body>
    <h2>Patient Registration Form</h2>
    <form action="process_registration.php" method="post">
        <label for="name">Full Name:</label>
        <input type="text" id="name" name="name" required><br
><br>

        <label for="dob">Date of Birth:</label>
        <input type="date" id="dob" name="dob" required><br><
br>

        <label for="address">Address:</label>
        <textarea id="address" name="address" rows="4" cols
="50" required></textarea><br><br>

        <!-- Add more fields as needed -->

        <input type="submit" value="Register">
    </form>
</body>
</html>
```

Step 2: Process Form Data and Insert into MySQL (process_registration.php)

```php
<?php
$servername = "localhost";
$username = "your_username";
$password = "your_password";
$dbname = "hospital_database";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbnam
e);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST['name'];
    $dob = $_POST['dob'];
    $address = $_POST['address'];

    // Perform data validation and sanitation here if needed

    // Insert data into the database
    $sql = "INSERT INTO patients (name, dob, address) VALUES
('$name', '$dob', '$address')";

    if ($conn->query($sql) === TRUE) {
        $message = "Registration successful!";
    } else {
        $message = "Error: " . $sql . "<br>" . $conn->error;
    }
}

// Close connection
$conn->close();
```

```
?>

<!DOCTYPE html>
<html>
<head>
    <title>Registration Result</title>
</head>
<body>
    <h2>Registration Result</h2>
    <p><?php echo $message; ?></p>
</body>
</html>
```

OR

(b) Differentiate between static and dynamic websites. Is PHP a case-sensitive scripting language?

Ans.

Feature	Static Website	Dynamic Website
Page Content	Fixed, hardcoded HTML	Generated dynamically by code
Interactivity	None	Highly interactive with forms, comments etc.
Customization	Same content for all users	Personalized content per user
Database connectivity	None	Connects to databases to load data
Development technology	Plain HTML, CSS, Javascript	Server-side languages like PHP, ASP.NET
Performance	Very fast since static files served	Can be slower due to page generation
Security	Generally more secure	Vulnerable to attacks by hackers
SEO	Easy to optimise pages	Can be more complex

Feature	Static Website	Dynamic Website
Use cases	Simple brochure sites	Web applications, e-commerce, social media

Regarding PHP, yes it is a case-sensitive language:

- Variable and function names depend on exact capitalisation
- Language keywords must be lowercase
- Built-in functions are lowercase like strlen(), array_push()
- Class, ID, and file names are case-sensitive

So case is significant when writing PHP code to avoid errors. Conventions like lower_case for variables and CamelCase for classes help keep things clear.

(c) What are the variable-naming rules you should follow in PHP? How are constants defined in PHP?

Ans.

some variable naming rules to follow in PHP:

- Variable names must start with a letter or underscore, not a number.
- Variable names can only contain alphanumeric characters and underscores. No special characters allowed.
- Variable names are case-sensitive ($name and $NAME are different variables).
- By convention, variables should be lowercase. Use underscores to separate words ($first_name).
- Avoid using PHP reserved words as variable names (if, echo, etc).
- Use meaningful variable names that describe the data they hold.
- Follow a consistent naming convention throughout your code.
- For global variables, prefix with g_ ($g_username).
- For class properties/members, prefix with m_ ($m_phone_number).

Constants in PHP are defined using the define() function:

- define('CONSTANT_NAME', value);
- CONSTANT_NAME must be in uppercase by convention.
- The value can be any type - string, number, boolean, etc.
- Constants cannot be changed after they are defined.
- Access constants without a dollar sign (echo CONSTANT_NAME;).

For example:

```
define('MAX_SIZE', 100);
echo MAX_SIZE;// 100
```

Constants provide immutable values and help avoid magic numbers in code.

Q 6. “Request and Response is the two essential objects in AJAX”. Take any scenario or example on your own and try to explain the roll of request and response object with code of AJAX using XML

Ans.

Let's consider a scenario where you want to fetch weather information for a specific city using AJAX (Asynchronous JavaScript and XML). In this scenario, the "Request" object is responsible for sending a request to a server to fetch data, and the "Response" object is used to handle the data received from the server.

Here's an example of how you might use AJAX to fetch weather information for a city using XML as the response format:

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Weather App</title>
    <script src="script.js"></script>
</head>
<body>
    <h1>Weather Information</h1>
```

```
    <label for="city">Enter City: </label>
    <input type="text" id="city" />
    <button id="getWeather">Get Weather</button>
    <div id="weatherInfo"></div>
</body>
</html>
```

JavaScript (`script.js`):

```
document.getElementById("getWeather").addEventListener("clic
k", function() {
    var city = document.getElementById("city").value;
    var url = "https://api.example.com/weather?city=" + city;

    var xhr = new XMLHttpRequest();
    xhr.open("GET", url, true);

    xhr.onreadystatechange = function() {
        if (xhr.readyState === 4 && xhr.status === 200) {
            var responseXML = xhr.responseXML; // Get the XML
response
            var temperature = responseXML.getElementsByTagNam
e("temperature")[0].textContent;
            var condition = responseXML.getElementsByTagName
("condition")[0].textContent;

            var weatherInfo = "Temperature: " + temperature +
"°C<br>Condition: " + condition;
            document.getElementById("weatherInfo").innerHTML
= weatherInfo;
        }
    };

    xhr.send();
});
```

In this example, the user enters a city name, and when they click the "Get Weather" button, an AJAX request is sent to the server. Let's assume the server responds with an XML document like this:

```
<weather>
    <temperature>25</temperature>
    <condition>Sunny</condition>
</weather>
```

The JavaScript code uses the XMLHttpRequest object to send a GET request to the server's API endpoint. The `onreadystatechange` event handler listens for changes in the request state. When the request is completed (readyState 4) and the status code is 200 (OK), it processes the XML response.

The XML response is parsed using `responseXML` property of the `xhr` object. The temperature and condition values are extracted from the XML using `getElementsByTagName()` and then displayed in the HTML.

In this scenario, the "Request" object (represented by the XMLHttpRequest) is responsible for making the API request, and the "Response" object (represented by the responseXML) is used to handle the XML data received from the server.

Q 7. “CSS and JavaScript both are used on Web pages with HTML but for different roles”. Justify the Statement. Develop an online application or webpage to find the transpose of the given matrix using JavaScript, input should be given by the user.

Ans.

You're absolutely right. CSS and JavaScript play very different roles when used together with HTML on web pages:

CSS (Cascading Style Sheets) is responsible for the visual styling and layout of web page content. Using CSS, you can control colors, fonts, positioning, animations, and the overall look and feel of the page. CSS is all about presentation.

JavaScript is a programming language that controls the dynamic behavior and interactivity of a web page. With JavaScript you can:

- Dynamically modify HTML and CSS to update content, styling, etc.
- Respond to user interactions like clicks, drags, key presses
- Validate form data and show error messages
- Make AJAX requests to get new data from a server without reloading the page
- Create animations and visual effects
- Add interactive elements like drop-down menus
- Build web/browser-based games and applications

So in summary:

- CSS handles styling and presentation
- JavaScript handles dynamic behavior and interactivity

HTML provides the basic semantic structure and content, CSS makes it look nice, and JavaScript makes it interactive and dynamic.

They complement each other with different responsibilities - CSS for looks, JavaScript for behavior and interactivity. Together they allow web pages to be highly interactive, responsive and usable for users.

So your statement is absolutely justified - CSS and JavaScript both work with HTML on web pages, but serve distinct purposes. CSS is for styling, JavaScript is for interactivity and dynamic effects. They work together to create great user experiences on the web!

```
<html>

<head>
    <script>
        function createMatrixInput() {
  var rows = document.getElementById("rows").value;
  var columns = document.getElementById("columns").value;
  var matrixTable = document.getElementById("matrixTable");
  matrixTable.innerHTML = "";
  for (var i = 0; i < rows; i++) {
    var row = document.createElement("tr");
    for (var j = 0; j < columns; j++) {
```

```
      var cell = document.createElement("td");
      var input = document.createElement("input");
      input.type = "number"
      input.id = "element_" + i + "_" + j;
      cell.appendChild(input);
      row.appendChild(cell);
    }
    matrixTable.appendChild(row);
  }
}

function calculateTranspose() {
  var rows = document.getElementById("rows").value;
  var columns = document.getElementById("columns").value;
  var matrix = [];
  for (var i = 0; i < rows; i++) {
    matrix[i] = [];
    for (var j = 0; j < columns; j++) {
      matrix[i][j] = document.getElementById("element_" + i + "_
    }
  }

  var transpose = [];
  for (var j = 0; j < columns; j++) {
    transpose[j] = [];
    for (var i = 0; i < rows; i++) {
      transpose[j][i] = matrix[i][j];
    }
  }
  var transposeTable = document.getElementById("transposeTable"
  transposeTable.innerHTML = "";
  for (var i = 0; i < columns; i++) {
    var row = document.createElement("tr");
    for (var j = 0; j < rows; j++) {
      var cell = document.createElement("td");
      cell.textContent = transpose[i][j];
```

```
            row.appendChild(cell);
        }
        transposeTable.appendChild(row);
    }
}
    </script>
</head>

<body>
    <input id="rows">
    <input id="columns">
    <button onclick="createMatrixInput()">Create Matrix</button>
    <button onclick="calculateTranspose()">Calculate Transpose</
    <table id="matrixTable">
    </table>
    <table id="transposeTable">
    </table>
    <script>
        window.onload = createMatrixInput;
    </script>
</body>

</html>
```

code explanation:

1. **HTML Structure:**
    - The code starts with the `<!DOCTYPE html>` declaration, indicating that this is an HTML5 document.
    - The `<head>` section includes the title of the webpage and the embedded CSS and JavaScript code.
    - The `<style>` tag contains CSS rules for styling the webpage, such as layout, fonts, and alignment.
    - The `<script>` tag includes JavaScript functions for creating the matrix input fields and calculating the transpose.

2. **JavaScript Functions:**
    - `createMatrixInput()`: This function is called when the "Create Matrix" button is clicked. It creates a table of input fields based on the specified number of rows and columns.
    - `calculateTranspose()`: This function is called when the "Calculate Transpose" button is clicked. It reads the matrix elements from the input fields, calculates the transpose, and displays it in a table.
    - The `window.onload` event listener ensures that the `createMatrixInput()` function is called when the page loads to initially create the matrix input fields.
3. **HTML Layout:**
    - The layout is designed using flexbox to center the content vertically and horizontally within the viewport.
    - The `<h1>` heading at the top displays the title of the webpage.
    - The matrix input section consists of input fields for the number of rows and columns, along with buttons to create the matrix and calculate the transpose.
    - The matrix input table and the transpose table are displayed as tables within their respective containers.
4. **Explanation of Flow:**
    - Users enter the number of rows and columns and click the "Create Matrix" button to generate the matrix input fields.
    - After entering matrix elements, users click the "Calculate Transpose" button to calculate and display the transpose.
    - The matrix input fields are created dynamically based on user input, and the transpose is calculated by rearranging the elements.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ac6a7de0-f5ba-4b33-b9e5-11b985e1266b/matrix.html

Q 8. “JSON is better than Xml”. Justify the statement and differentiate with suitable example or code. Explain the uses of JQuery.

Ans.

Feature	JSON	XML
Structure	Uses key-value pairs inside curly braces `{ }`	Uses tags to markup data `<note> </note>`
Example	`{ "name": "John", "age": 30 }`	`<note> <to>John</to> <from>Jane</from> </note>`
Data Types	Supports strings, numbers, booleans, null	Supports strings, dates, numbers
Readability	More readable and compact	Verbose, more difficult to read
Parsing	Faster to parse in most programming languages	Slower to parse due to verbosity
Usage	Mainly for data exchange and storage	Wide range of uses including configuration, documents

In summary, JSON is better for data exchange and storage as it is more lightweight and faster to parse. XML is more verbose but has a wider range of use cases like configuration files and documents.

JQuery is a JavaScript library that makes it easier to interact with and manipulate the DOM (Document Object Model). Here are some key uses of JQuery:

- DOM manipulation - Makes it easy to select, create, remove, traverse DOM elements
- Event handling - Simple ways to attach event listeners and respond to events
- AJAX/API calls - Helper functions to make AJAX requests and handle responses
- Animations/Effects - Animate/toggle CSS properties, show/hide elements
- Cross-browser support - Normalizes behavior across browsers
- Plugin architecture - Easy to create and use plugins for added functionality
- Lightweight - Much lighter than other JavaScript frameworks at only ~100KB

So in summary, JQuery makes client-side scripting easier with its concise syntax and cross-browser compatibility. It streamlines a lot of common JavaScript tasks for interacting with the DOM, animations, AJAX calls etc. Many websites still use JQuery today as a lightweight supplement to frameworks like React.

Q 9. How do the table tags with the different attributes helps in designing the webpage? How do we create the registration form in the webpage with help of table, write the code?

Ans.

The table tags along with attributes like border, cellpadding, cellspacing, width, etc. help in designing webpages in several ways:

- They provide structure and layout to content on the page. The tabular format helps organize content in rows and columns.
- Borders and cellpadding attributes can visually separate and style different sections of content.
- Width can control the overall width of the table and size it appropriately for the page.
- Cellspacing adjusts spacing between cells and fine tunes the look.
- Attributes like colspan and rowspan allow spanning columns and rows to create more complex layouts.
- Align attributes can align text and elements within table cells.
- Thead, tbody and tfoot separate header, body and footer sections visually.
- Summary and caption can add additional descriptive text.
- Tables also provide semantic structure to data like charts, calendars, pricing plans etc.

Registration form:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Registration Form</title>
</head>

<body>

<h2>Registration Form</h2>

<table border="1" cellpadding="5">

  <tr>
    <td>First Name:</td>
    <td><input type="text" name="first_name"></td>
  </tr>

  <tr>
    <td>Last Name:</td>
    <td><input type="text" name="last_name"></td>
  </tr>

  <tr>
    <td>Email:</td>
    <td><input type="email" name="email"></td>
  </tr>

  <tr>
    <td>Password:</td>
    <td><input type="password" name="password"></td>
  </tr>

  <tr>
    <td>Gender:</td>
    <td>
      <input type="radio" name="gender" value="male"> Male
      <input type="radio" name="gender" value="female"> Fe
male
```

```
    </td>
  </tr>

  <tr>
    <td>Phone Number:</td>
    <td><input type="text" name="phone"></td>
  </tr>

  <tr>
    <td>Address:</td>
    <td><textarea name="address"></textarea></td>
  </tr>

  <tr>
    <td colspan="2" align="center">
      <input type="submit" value="Register">
    </td>
  </tr>

</table>

</body>
</html>
```

The key table attributes used here are:

- border - Adds visible border around table and cells
- cellpadding - Adds padding inside each cell
- colspan - Merges the submit button column spanning two columns
- align - Aligns the submit button to the center

Q 10. How DOM (Document Object Model) help to create webpage? How to find and access HTML elements in an HTML page.

Ans.

The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM provides a structured representation of the document as a tree of nodes and objects that can be accessed and manipulated.

Here are a few ways the DOM helps create and interact with webpages:

- The DOM defines a standard for accessing and manipulating documents. This allows programs to be written that work across different browsers.
- It provides APIs and methods for creating, removing and modifying page elements, attributes, styles, etc. For example, you can create a new div, set its text content, add a class, etc.
- It represents the document as a structured tree. This allows you to "traverse" the tree and find specific elements, access parent/child relationships, and more.
- It handles events and user interaction. The DOM allows attaching event listeners and handlers to nodes.
- It defines the logical structure of documents and the way a document is accessed and manipulated. This "web page interface" provides another layer on top of the raw HTML.

Some common ways to access and manipulate the DOM include:

- getElementById() - access an element by its ID
- getElementsByClassName() - access elements by their class name
- querySelector() - find the first matching element
- parentNode - access an element's parent node
- childNodes - access an element's child nodes
- innerHTML - access or modify an element's inner HTML
- textContent - access or modify the text content of an element
- createElement() - create a new element
- appendChild() - add a new child element

- addEventListener() - attach an event handler
- removeChild() - remove an element

So in summary, the DOM provides a standard programming interface for HTML and XML documents that allows manipulating the document structure, styling, and content through a logical tree representation.