Web Technology Oral Questions and Answers

1. Q: What is HTML?

A: HTML (HyperText Markup Language) is the standard language used to create and design documents on the World Wide Web. It uses a series of elements and tags to structure content, such as headings, paragraphs, links, images, and multimedia elements, which are then rendered by web browsers.

2. Q: What are Tags?

A: Tags are special keywords enclosed in angle brackets used in HTML to define elements on a web page. They generally appear in pairs, with an opening and closing tag, to enclose content. Tags specify formatting, layout, and structure of the document elements such as text, images, and links.

3. Q: Do all HTML tags have end tag?

A: No, not all HTML tags have an end tag. Some tags are self-closing or void elements, meaning they don't require a closing counterpart. Examples include
, <img>, and <hr>. These tags do not wrap content and serve specific purposes like inserting line breaks or displaying images.

4. Q: What is the difference between HTML elements and tags?

A: Tags are the codes enclosed in angle brackets like or . HTML elements, on the other hand, consist of the tags and the content within them. For example, Hello is a paragraph element where and are tags and 'Hello' is the content.

5. Q: How to insert a copyright symbol on a browser page?

A: To insert a copyright symbol in HTML, use the entity code ©. When rendered in a browser, this code will display the copyright symbol ©. It is commonly used in the footer of web pages to denote intellectual property ownership for content or designs on the page.

6. Q: How do you keep list elements straight in an HTML file?

A: List elements can be structured using <ul> for unordered lists or <ol> for ordered lists, with each item enclosed in <li> tags. Proper indentation and nesting ensure readability and a clean structure. CSS can also be used to control spacing, alignment, and overall appearance of list elements.

Q7: Does a hyperlink only apply to text?
A: No, a hyperlink in HTML can be applied to various elements, not just text. It can link

images, buttons, or even sections of a page. The <a> tag allows any nested HTML content to act as a clickable hyperlink to internal or external web addresses.

Q8: What is a style sheet?
A: A style sheet is a set of rules in CSS that define how HTML elements should appear in a web page. It controls the layout, fonts, colors, spacing, and overall design. Style sheets can be internal, external, or inline and help separate content from presentation.

Q9: Can you create a multi colored text on a web page?
A: Yes, using CSS, you can apply different colors to individual parts of text. This can be done by wrapping text in tags and assigning each a class or style. For example: Red Blue displays multi-colored text.

Q10: Is it possible to change the color of the bullet?
A: Yes, CSS allows customization of list bullets. You can use the ::marker pseudo-element or apply color to the entire list item and use custom images as bullets. This enables you to match bullet points with the website's color scheme and design aesthetics.

Q11: What is a marquee?
A: A marquee is an HTML element <marquee> used to create scrolling text or images horizontally or vertically across the screen. Although once popular, it's now obsolete and not recommended in modern web development due to poor accessibility and lack of support in HTML5.

Q12: How many tags can be used to separate sections of texts?
A: Several HTML tags can separate text sections, such as , <section>, <article>, <header>, <footer>, <aside>, and <p>. These tags structure the content semantically and visually, allowing browsers and developers to understand the page layout and content blocks more effectively.

Q13: How to make a picture a background image of a web page?
A: Use CSS to set a background image for the body or any container. Example:

css

```
CopyEdit

body {

  background-image: url('image.jpg');

  background-size: cover;

  background-repeat: no-repeat;

}
```

This applies the image as the page's background, often for design or branding.

Q14: What is the use of iframe tag?
A: The <iframe> tag embeds another HTML page within the current page. It's useful for displaying external content like maps, videos, ads, or entire websites without redirecting users. However, care must be taken due to potential security and cross-origin issues.

Q15: What are the different new form element types in HTML 5?
A: HTML5 introduced new input types like email, url, date, range, search, color, tel, and datetime-local. These improve form usability, validation, and mobile experience by automatically providing appropriate keyboards and validation mechanisms in modern browsers.

Q16: Is there any need to change the web browsers to support HTML5?
A: No. Most modern web browsers including Chrome, Firefox, Safari, Edge, and Opera support HTML5 natively. Older versions of Internet Explorer may require polyfills or fallback solutions. HTML5 was designed to be backward compatible and progressively enhance user experience.

Q17: What is XML?
A: XML (eXtensible Markup Language) is used to store and transport data. Unlike HTML, it doesn’t define presentation but focuses on the structure of data. It's both human- and machine-readable and allows users to create custom tags for organizing and transmitting information across different platforms.

Q18: What are the features of XML?
A: XML is platform-independent, self-descriptive, supports Unicode, and is both human- and machine-readable. It allows hierarchical data structuring, data validation via DTD/XSD, and separates data from presentation. It also supports nesting, extensibility, and can be easily integrated with other technologies.

Q19: What are the differences between HTML and XML?
A: HTML focuses on how data looks, while XML focuses on what data is. HTML uses predefined tags; XML allows custom tags. HTML is forgiving with errors; XML is strict. XML is used for transporting and storing data; HTML is for displaying data in browsers.

Q20: What is XML DOM Document?
A: The XML DOM (Document Object Model) is a programming interface that treats an XML document as a tree structure, where each element is an object. It allows developers to dynamically access and manipulate the content, structure, and style of XML documents using languages like JavaScript.

Q21: What is an attribute?
A: An attribute in XML provides additional information about elements. It appears inside the opening tag and consists of a name-value pair. Example: <person age="25">. Attributes should be used sparingly, especially when the information can be represented as sub-elements.

Q22: What are the advantages of XML schema over DOM Document?
A: XML Schema defines the structure and data types of XML documents, allowing strict validation. It supports data types, namespaces, and complex structures better than DTD. Unlike DOM, it doesn't represent document content, but ensures correctness before DOM is parsed or manipulated.

Q23: What are the basic rules while writing XML?
A: XML must have a single root element. Tags must be properly nested and closed. Attribute values must be quoted. It is case-sensitive, and names cannot start with numbers. Every opening tag must have a corresponding closing tag unless it's self-closing.

Q24: What is XML Element?
A: XML elements are the primary building blocks of XML. They consist of a start tag, content, and end tag. For example, <name>John</name>. Elements can also nest other elements, forming a tree-like structure. They define the actual data and structure of XML documents.

Q25: What is CDATA?
A: CDATA (Character Data) section is used in XML to include content that should not be parsed by the XML parser. Anything inside <![CDATA[...]]> is treated as literal text, allowing characters like < or & without needing to escape them.

Q26: How comment can be represented in XML?
A: Comments in XML are written using the syntax <!-- comment goes here -->. They can span multiple lines and are ignored by XML parsers. Comments are useful for explanations or annotations but should not be used inside tags or attributes.

Q27: What are XML Namespaces?
A: XML Namespaces prevent element name conflicts in documents that mix elements from

different XML vocabularies. They are defined using the xmlns attribute and typically use a URL or URI to uniquely identify the namespace.

Q28: What is XSL?
A: XSL (eXtensible Stylesheet Language) is used to transform and style XML documents. XSLT (XSL Transformations) is a key component that converts XML into other formats like HTML, plain text, or other XML formats. It supports conditional formatting and loops.

Q29: What is XML Schema?
A: XML Schema (XSD) defines the structure, content, and data types of XML documents. It validates XML data for correctness. XSD is more powerful than DTD and supports namespaces, data typing, inheritance, and constraints like min/max values or patterns.

Q30: What is CSS?
A: CSS (Cascading Style Sheets) is used to describe how HTML elements should be displayed. It controls layout, colors, fonts, spacing, and more. CSS separates content from presentation, allowing consistent styling across multiple pages and better maintainability in modern web development.

Q31: What are advantages of using CSS?
A: CSS reduces repetition by allowing centralized style control, improves website speed by separating design from HTML, enhances accessibility, ensures consistent layout, and simplifies maintenance. It also supports responsive design, animations, and a cleaner HTML structure.

Q32: What are the components of a CSS Style?
A: A CSS style rule consists of a selector and a declaration block. The selector targets HTML elements, and the declaration block contains one or more property-value pairs (e.g., color: red;). Example: p { font-size: 16px; color: blue; }.

Q33: What is type selector?
A: A type selector in CSS targets HTML elements by their tag name. For example, h1 { color: green; } selects all <h1> elements and applies the defined styles to them.

Q34: What is universal selector?
A: The universal selector * matches every element on the page. For example, * { margin: 0; padding: 0; } resets all default spacing for all elements. It's useful for applying global styles.

Q35: What is Descendant Selector?
A: A descendant selector targets elements that are nested within a specified parent. For example, div p applies to all <p> tags inside a . It allows targeted styling based on HTML structure.

Q36: What is class selector?
A: A class selector targets elements with a specific class attribute. It's defined with a period

prefix. For example, .highlight { color: red; } applies styles to all elements with class="highlight".

Q37: Can you make a class selector particular to an element type?
A: Yes, you can specify both the tag and class. For example, p.notice applies styles only to <p> elements with class="notice", not to other elements with the same class.

Q38: What is id selector?
A: An ID selector targets a specific element using the id attribute. It is denoted with a hash symbol, like #header. IDs must be unique on a page, unlike classes which can be reused.

Q39: Can you make an id selector particular to an element type?
A: Yes. For example, div#main will apply styles only to a with id="main". This helps ensure styling is limited to a specific tag with a specific ID.

Q40: What is a child selector?
A: A child selector applies styles to direct children of an element. Syntax: parent > child. For example, ul > li styles only the immediate <li> items inside <ul>, not nested s.

Q41: What is an attribute selector?
A: Attribute selectors target elements based on attributes and values. For example, input[type="text"] styles only <input> elements where type is "text". They are useful for form styling and accessibility enhancements.

Q42: How to select all paragraph elements with a lang attribute
A: Use the CSS selector p[lang] to target all <p> elements that include a lang attribute, regardless of its value. This is useful for language-specific styling or localization features.

Q43: Explain types of CSS
A: CSS has three types: Inline (written inside the HTML tag), Internal (within <style> tags in the HTML head), and External (in a separate .css file). External is preferred for scalability and reuse across pages.

Q44: Explain Internal CSS, External CSS, Inline CSS
A: Inline CSS is added directly to an HTML element using the style attribute. Internal CSS is placed within <style> tags in the HTML file's <head>. External CSS is stored in a separate file and linked via <link>—ideal for styling multiple pages.

Q45: What is JSON?
A: JSON (JavaScript Object Notation) is a lightweight data-interchange format used for storing and transferring data. It's human-readable and easy to parse, widely used in web APIs for communication between client and server.

Q46: Explain what are JSON objects?
A: JSON objects are unordered collections of key-value pairs, enclosed in curly braces. Each key is a string, and each value can be a string, number, array, boolean, null, or another object. Example: { "name": "Alice", "age": 25 }.

Q47: Explain how to transform JSON text to a JavaScript object?
A: Use JSON.parse() method in JavaScript. Example:

javascript

CopyEdit

let obj = JSON.parse('{"name": "Alice"}');

This converts a JSON string into a usable JavaScript object with properties accessible via dot or bracket notation.

Q48: Why must one use JSON over XML?
A: JSON is more lightweight, easier to read/write, and integrates naturally with JavaScript. It has less overhead and supports modern data formats. It's faster to parse and typically smaller in size compared to verbose XML documents.

Q49: Mention what is the file extension of JSON?
A: The standard file extension for JSON files is .json. These files typically contain structured data used for configuration, data exchange, and APIs.

Q50: Mention which function is used to convert a JSON text into an object?
A: JSON.parse() is used in JavaScript to convert a JSON string into a JavaScript object, allowing the data to be manipulated or displayed dynamically.

Q51: Mention what are the data types supported by JSON?
A: JSON supports six data types: string, number, boolean, null, array, and object. These types cover most data representation needs and align closely with JavaScript's types for seamless integration.

Q1: What is JavaScript?
A: JavaScript is a high-level, interpreted programming language used to create dynamic and interactive content on websites. It runs in the browser and can manipulate the DOM, respond to user actions, validate forms, and communicate with servers using AJAX, enhancing web application interactivity.

Q2: What is the difference between JavaScript and JScript?
A: JavaScript is the standard scripting language used in browsers, while JScript is Microsoft's proprietary version of JavaScript introduced in Internet Explorer. Both are based on ECMAScript, but JScript includes Windows-specific features and was mainly supported by older versions of Internet Explorer.

Q3: How to write a hello world example of JavaScript?
A: A simple Hello World script can be written like:

html

CopyEdit

<script>

alert("Hello, World!");

</script>

This code, placed in the HTML body or head, shows a popup message. Alternatively, console.log("Hello, World!") prints the message to the browser console.

Q4: How to use external JavaScript file?
A: Save JavaScript code in a .js file and include it in the HTML using:

html

CopyEdit

<script src="script.js"></script>

This improves modularity and keeps HTML clean. The browser loads and executes the script from the external file when the page loads.

Q5: Is JavaScript case sensitive language?
A: Yes, JavaScript is case-sensitive. Variable names, function names, and keywords must match in case. For example, myVar and myvar are treated as two different variables. Improper casing results in errors or unexpected behavior.

Q6: What is DOM? What is the use of document object?
A: The DOM (Document Object Model) represents HTML as a tree of objects. The document object provides access to page elements and allows manipulation through JavaScript. It enables developers to dynamically change content, style, and structure of a web page.

Q7: What is the use of window object?
A: The window object is the global object representing the browser window. It provides functions and properties for controlling the browser (like alert(), setTimeout()), accessing the document, location, history, and more. All global variables are part of the window object.

Q8: How to write comment in JavaScript?
A: JavaScript supports two types of comments:

- Single-line: // comment
- Multi-line:

javascript

CopyEdit

/* comment block */

Comments help document the code and are ignored during execution.

Q9: How to create function in JavaScript?
A: A function is defined using:

javascript

CopyEdit

function greet() {

alert("Hello!");

}

You can also use function expressions or arrow functions for compact syntax:

javascript

CopyEdit

const greet = () => alert("Hello!");

Q10: What are the JavaScript data types?
A: JavaScript supports the following data types: string, number, boolean, null, undefined, symbol, bigint (primitive types), and object (complex type including arrays and functions). It is dynamically typed, so variables can hold different types over time.

Q11: How to create objects in JavaScript?
A: You can create objects using:

javascript

CopyEdit

let obj = { name: "Alice", age: 25 };

Or with the constructor:

javascript

CopyEdit

let obj = new Object();

obj.name = "Bob";

Objects store data as key-value pairs.

Q12: How to create array in JavaScript?
A: Arrays can be created using:

javascript

CopyEdit

let arr = [1, 2, 3];

Or with the constructor:

javascript

CopyEdit

let arr = new Array("a", "b");

Arrays are ordered lists of values accessible by index.

Q13: Difference between Client side JavaScript and Server side JavaScript?
A: Client-side JavaScript runs in the browser and handles UI interaction, form validation, etc. Server-side JavaScript (e.g., Node.js) runs on the server, handles databases, APIs, and file systems. They differ in environment, capabilities, and purposes.

Q14: In which location cookies are stored on the hard disk?
A: Cookies are stored in the browser's profile or user data folder on the hard disk. Their exact location varies by browser and operating system. They are tied to specific domains and accessed automatically by the browser during relevant requests.

Q15: What is the difference between Internal & External Javascript.
A: Internal JavaScript is written inside <script> tags within the HTML file. External JavaScript is stored in a separate .js file and linked with the src attribute. External scripts promote reusability and cleaner HTML structure.

---

jQuery Questions

Q16: What is jQuery?
A: jQuery is a fast, lightweight JavaScript library designed to simplify HTML DOM manipulation, event handling, animation, and AJAX. It provides cross-browser compatibility and concise syntax, allowing developers to write less code for common JavaScript tasks.

Q17: Why do we use jQuery?
A: jQuery simplifies client-side scripting by offering an easy-to-use API for DOM manipulation, event handling, animations, and AJAX. It reduces code complexity and ensures compatibility across browsers, speeding up development and improving maintainability.

Q18: How JavaScript and jQuery are different?
A: JavaScript is a core programming language, while jQuery is a library built on top of JavaScript. jQuery simplifies JavaScript tasks using concise syntax and handles cross-browser issues, whereas raw JavaScript offers full control but with more code.

Q19: Is jQuery replacement of Java Script?
A: No, jQuery is not a replacement for JavaScript. It's a library that makes certain tasks easier in JavaScript. JavaScript provides the foundation, and jQuery builds upon it to simplify DOM manipulation, event handling, and animations.

Q20: Is jQuery a W3C standard?
A: No, jQuery is not a W3C standard. It is an independent, open-source JavaScript library developed by John Resig. However, it uses W3C-standard-compliant JavaScript and HTML features.

Q21: What is the basic need to start with jQuery?
A: To use jQuery, you include the jQuery library in your HTML file via a local file or a CDN link. Then, you can write jQuery code inside $(document).ready() to ensure the DOM is fully loaded before executing scripts.

Q1: How many objects of a servlet is created?
A: Only one object of a servlet is typically created per servlet class by the servlet container. This single instance handles all client requests concurrently using multiple threads. This ensures efficient memory usage and supports concurrent user interactions.

Q2: What is the life-cycle of a servlet?
A: The servlet lifecycle includes:

1. **Loading** and instantiation
2. **Initialization** via init()
3. **Request handling** via service()
4. **Destruction** via destroy()
   The servlet container manages this lifecycle to ensure proper resource management and request handling.

Q3: What are the life-cycle methods for a servlet?
A: The three core lifecycle methods are:

- init() – called once for initialization
- service() – called for each request
- destroy() – called before servlet is removed
  These methods allow setup, execution, and cleanup within the servlet.

Q4: Who is responsible to create the object of servlet?
A: The servlet container (e.g., Tomcat) is responsible for creating and managing the servlet object. It ensures thread safety, manages initialization, and invokes lifecycle methods as needed for request processing.

Q5: When servlet object is created?
A: The servlet object is created when the servlet is first requested or when the server starts (if load-on-startup is configured). It is instantiated once and reused for handling multiple requests.

Q6: What is difference between Get and Post method?
A: GET appends data to the URL and is visible; it is suitable for fetching data. POST sends data in the request body and is used for submitting forms. POST is more secure and can handle larger data volumes.

Q7: What is difference between PrintWriter and ServletOutputStream?
A: PrintWriter is used to send character data (text) in the response, while ServletOutputStream sends binary data (like images or PDFs). Use PrintWriter for HTML output and ServletOutputStream for media or downloadable content.

Q8: What is difference between GenericServlet and HttpServlet?
A: GenericServlet is protocol-independent and defines a base class for all servlets. HttpServlet extends GenericServlet and adds HTTP-specific methods like doGet() and doPost(). Most web applications use HttpServlet for HTTP-based communication.

Q9: What is Session Tracking?
A: Session tracking is a way to maintain user state and data across multiple HTTP requests. It can be implemented using cookies, URL rewriting, hidden form fields, or the HttpSession API in Java servlets.

Q10: What are Cookies?
A: Cookies are small pieces of data stored on the client-side by the browser. They help maintain session information, track user activity, and personalize user experience. Servlets can create, read, and delete cookies using the javax.servlet.http.Cookie class.

Q11: What is difference between Cookies and HttpSession?
A: Cookies store data on the client side and persist between sessions. HttpSession stores data on the server side and provides better security. Sessions are ideal for sensitive or temporary data, while cookies are used for lightweight, long-term storage.

JSP Questions

Q12: What is JSP?
A: JSP (JavaServer Pages) is a technology for developing dynamic web pages using HTML

and embedded Java code. It allows mixing static content with Java logic and is compiled into a servlet behind the scenes by the server.

Q13: What are the life-cycle methods for a JSP?
A: JSP lifecycle includes:

1. **Translation** to servlet
2. **Compilation**
3. **Initialization** via jspInit()
4. **Request handling** via _jspService()
5. **Destruction** via jspDestroy()
   The servlet container manages this lifecycle automatically.

Q14: What is difference between hide comment and output comment?
A: JSP hide comments (<%-- comment --%>) are not sent to the browser and remain server-side. HTML output comments (<!-- comment -->) are visible in the browser's page source. Hide comments are useful for internal documentation.

Q15: What are the JSP implicit objects?
A: JSP provides implicit objects like request, response, session, application, out, config, pageContext, page, and exception. These are automatically available in JSP pages for handling user input, output, session data, and context-related tasks.

Q16: What are the 3 tags used in JSP bean development?
A: The three tags are:

- <jsp:useBean> – instantiates or locates a bean
- <jsp:setProperty> – sets bean properties
- <jsp:getProperty> – retrieves and displays property values
  These tags enable integration of JavaBeans in JSP.

Q17: What are the advantages of JSP over JavaScript?
A: JSP runs on the server and can connect to databases, generate dynamic content, and secure business logic. JavaScript runs in the browser and is limited to client-side tasks. JSP is better for dynamic server-side processing.

Q18: What is a scriptlet in JSP and what is its syntax?
A: A scriptlet is a block of Java code embedded within a JSP page using <% ... %>. It executes on the server and can contain variables, logic, or method calls. Example: <% int x = 5; %>.

Q19: What are JSP declarations?
A: Declarations allow defining methods or variables at the class level in JSP using <%! ... %>.

These are outside the service method and shared across requests. Example: <%! int count = 0; %> defines a class-level variable.

Q20: What are JSP expressions?
A: JSP expressions output values directly into the HTML using <%= expression %>. Example: <%= 2 + 2 %> renders 4 on the web page. They are evaluated and converted to strings during request processing.

Q21: What are JSP comments?
A: JSP comments are written as <%-- comment --%>. They are not included in the HTML sent to the client, making them ideal for hiding server-side notes or logic that should not be visible in the browser.

Q22: What are JSP Directives?
A: JSP directives provide global information about the page and influence how it's processed. They are enclosed within <%@ ... %> and include page, include, and taglib. Directives must appear at the top of the page.

Q23: What are the types of directive tags?
A: JSP has three directive tags:

- <%@ page ... %> – defines page settings
- <%@ include ... %> – includes static resources
- <%@ taglib ... %> – imports custom tag libraries

Q24: What are JSP actions?
A: JSP action tags are XML syntax tags like <jsp:include>, <jsp:forward>, and <jsp:useBean>. They control the behavior of the JSP engine and interact with other components such as JavaBeans and external resources.

Q25: What is a page directive?
A: The page directive provides configuration settings for the JSP page. Example: <%@ page language="java" contentType="text/html" %>. It defines scripting language, content type, error pages, and more.

Q26: What is JSP Scripting Element.
A: JSP scripting elements are code blocks written in Java inside a JSP file. They include declarations <%! %>, scriptlets <% %>, and expressions <%= %>. These allow embedding server-side logic directly within the HTML.

Q27: Difference between JSP and Servlet.
A: JSP is a view technology designed for UI and easier content creation, while servlets are for logic processing and controlling flow. JSP is compiled into a servlet, but servlets require more boilerplate code. Typically, servlets handle backend logic and JSPs handle presentation.

1. What is PHP?
PHP (Hypertext Preprocessor) is a server-side scripting language used to develop dynamic web pages and web applications. It is embedded within HTML and executed on the server, generating HTML that is sent to the client's browser.

2. What does the initials of PHP stand for?
Originally "Personal Home Page", PHP now stands for "PHP: Hypertext Preprocessor", a recursive acronym. It highlights PHP's role in processing server-side scripts to dynamically generate web content.

3. Which programming language does PHP resemble to?
PHP syntax resembles C, Java, and Perl. It uses C-like syntax structures, control flow, and function handling, making it familiar to developers with backgrounds in those languages.

4. Differences between GET and POST methods?
GET appends data to the URL, visible and limited in size. POST sends data in the body, more secure and allows larger payloads. GET is used for retrieval; POST for submitting data.

5. How to declare an array in PHP?
Use the array() function or short syntax:

php

CopyEdit

$colors = array("red", "green");

$fruits = ["apple", "banana"];

Arrays store multiple values in one variable.

6. What is use of in_array() function in PHP?
in_array() checks if a value exists in an array.

php

CopyEdit

if (in_array("apple", $fruits))

Returns true if found. Useful for validation or filtering.

7. What is use of count() function in PHP?
count() returns the number of elements in an array.

php

CopyEdit

$count = count($fruits);

It's used to iterate or validate arrays.

8. What's the difference between include and require?
Both include files, but require causes a fatal error if the file is missing, while include only throws a warning and continues script execution.

9. What is the difference between Session and Cookie?
Cookies store data on the client side; sessions store data on the server. Cookies persist beyond sessions, while sessions expire when the browser is closed or after timeout.

10. How to set cookies in PHP?
Use setcookie("name", "value", time()+3600);. This sets a cookie valid for 1 hour. Must be used before any output is sent to the browser.

11. How to Retrieve a Cookie Value?
Access cookies using the $_COOKIE superglobal.

php

CopyEdit

echo $_COOKIE["name"];

Check if it exists using isset() to avoid errors.

12. How to create a session? How to set a value in session? How to Remove data from a session?
Start session with session_start();.
Set: $_SESSION['user'] = "John";
Remove: unset($_SESSION['user']);
Destroy all: session_destroy();

13. What types of loops exist in PHP?
PHP supports for, while, do-while, and foreach loops. foreach is ideal for iterating over arrays. Each serves specific control needs.

14. What is the difference between PHP and JavaScript?
PHP is server-side, JavaScript is client-side. PHP runs on the server and returns HTML, JavaScript manipulates HTML in the browser. They complement each other in full-stack web apps.

15. What is the PHP MySQLi and PHP PDO difference with example?
MySQLi is MySQL-specific; PDO supports multiple databases.
PDO:

php

CopyEdit

$db = new PDO(...);

MySQLi:

php

CopyEdit

$conn = new mysqli(...);

PDO is more flexible and secure.

16. What is PHP Superglobals?
Superglobals are built-in variables accessible globally, such as $_POST, $_GET, $_SESSION, $_COOKIE, $_SERVER, and $_FILES. They handle input, server info, and user session data.

AJAX Questions

1. What is AJAX?
AJAX (Asynchronous JavaScript and XML) allows web pages to update data without reloading. It communicates with the server in the background using JavaScript and XMLHttpRequest.

2. What are the advantages of AJAX?
Improves user experience by avoiding full page reloads, saves bandwidth, enhances responsiveness, and enables dynamic content loading like form submission, live search, and auto-refresh.

3. What are the disadvantages of AJAX?
AJAX can break browser history, be less accessible, and not work without JavaScript. It complicates SEO and increases development complexity due to async behavior.

4. Real web applications of AJAX currently running in the market?
Gmail, Google Maps, Facebook chat, Twitter feeds, and dynamic shopping carts use AJAX for real-time updates and seamless user interaction.

5. What are the security issues with AJAX?
AJAX faces CSRF, XSS, and data exposure risks. Proper validation, authentication, and HTTPS must be used to secure AJAX applications.

6. What is the difference between synchronous and asynchronous requests?
Synchronous requests block further code execution until response. Asynchronous lets other scripts run while waiting for the server's response, improving performance.

7. What are the technologies used by AJAX?
AJAX uses JavaScript, XMLHttpRequest, HTML, CSS, DOM, JSON/XML, and sometimes jQuery or fetch API for cross-browser compatibility and simplicity.

8. What does XMLHttpRequest?
It's an API used to send HTTP requests and receive responses asynchronously. It supports GET, POST, and can handle XML/JSON/HTML responses from the server.

9. What are the properties of XMLHttpRequest?
Important properties include readyState, status, responseText, and responseXML. They provide details about request progress and server response.

10. What are the important methods of XMLHttpRequest?
Key methods:

- open() – initializes request
- send() – sends request
- setRequestHeader() – sets headers
- abort() – cancels request

11. What are the different ready states of a request in AJAX?
0 = UNSENT, 1 = OPENED, 2 = HEADERS_RECEIVED, 3 = LOADING, 4 = DONE. The final state (4) means the operation is complete.

12. What are the common AJAX frameworks?
jQuery, Axios, Angular, React, Vue.js, and Prototype.js are popular AJAX-supporting frameworks for easier and cleaner code.

13. What is the difference between JavaScript and AJAX?
JavaScript is a full scripting language; AJAX is a technique using JavaScript to send async requests. AJAX relies on JavaScript and browser APIs.

AngularJS Questions

1. What is AngularJS?
AngularJS is a JavaScript framework by Google used to build dynamic single-page applications (SPAs). It extends HTML with new attributes and binds data seamlessly.

2. What are the advantages of AngularJS?
Two-way data binding, modular code structure, dependency injection, built-in directives, and testability are key benefits. It simplifies dynamic UI development.

3. What are the disadvantages of AngularJS?
Steep learning curve, performance issues with large DOMs, and being outdated compared to Angular (2+) are its cons. It also relies heavily on JavaScript.

4. What IDE's are used for AngularJS development?
Popular IDEs include Visual Studio Code, WebStorm, Sublime Text, Atom, and Eclipse. These provide syntax highlighting, auto-completion, and debugging support.

5. What are the features of AngularJS?
Features include two-way data binding, MVC pattern, dependency injection, custom directives, templating, and filters for formatting data.

6. What are directives in AngularJS?
Directives are markers on elements (like ng-model) that tell AngularJS to attach specific behaviors or transform the DOM.

7. What are services in AngularJS?
Services are reusable singleton objects that provide functionality like HTTP requests, logging, or shared data. Built-in services include $http, $timeout.

8. What is scope in AngularJS?
Scope is an object that binds the controller and view. It holds model data and watches for changes to update the UI automatically.

9. What is template in AngularJS?
A template is HTML with Angular-specific markup like {{expression}}. It defines the UI and binds it to controller logic.

10. What are expressions in AngularJS?
Expressions evaluate and display dynamic values in the view, enclosed in double curly braces ({{ }}). They support operations and functions.

11. What is the use of filter in AngularJS?
Filters format data in the view. Examples: uppercase, currency, date, filter. They can be custom or built-in.

12. What is MVC?
MVC stands for Model-View-Controller. It separates application data (Model), UI (View), and logic (Controller) for modularity and scalability.

13. Explain AngularJS Events.
AngularJS provides event directives like ng-click, ng-submit to bind events in HTML to controller functions, enabling interactive applications.

Q1: What is Node.js?
Node.js is an open-source, cross-platform runtime environment that executes JavaScript code outside a browser. Built on Chrome’s V8 engine, it enables developers to use JavaScript

for server-side scripting, handling I/O operations, file systems, and networking, allowing full-stack JavaScript development and building scalable network applications.

Q2: How Node.js works?
Node.js uses an event-driven, non-blocking I/O model. It runs on a single-threaded event loop that offloads operations like file and network calls to the system kernel or thread pool. Callbacks, Promises, or async/await handle asynchronous results, enabling high concurrency without spawning new threads for each connection.

Q3: What do you mean by the term I/O?
I/O (Input/Output) refers to operations where a system reads data from input sources (files, network requests, user input) or writes data to output destinations (databases, network responses, consoles). Efficient I/O handling is critical for server performance, especially in high-traffic or data-intensive applications.

Q4: What is Sturt?
There is no standard Node.js concept named "Sturt." You might mean **streams**, which handle data in chunks to process large datasets efficiently. Streams (Readable, Writable, Duplex, Transform) enable piping and incremental data handling without loading everything into memory at once, improving performance and resource usage.

Q5: What does event-driven programming mean?
Event-driven programming is a paradigm where the flow of the program is determined by events such as user actions or I/O completions. Handlers (callbacks) are registered for specific events, and when those events occur, the runtime invokes the corresponding callback, enabling asynchronous, non-blocking code execution.

Q6: Where can we use Node.js?
Node.js is ideal for real-time applications (chat, streaming), RESTful APIs, microservices, command-line tools, and server-side rendering. Its non-blocking I/O suits e-commerce, IoT, and single-page applications, providing high concurrency and fast response times.

Q7: What is the advantage of using Node.js?
Node.js offers high performance via non-blocking I/O and its event loop, enabling many concurrent connections. JavaScript on both client and server reduces context switching. With its rich npm ecosystem, Node.js accelerates development of scalable, maintainable network and web applications.

Q8: What are the two types of API functions in Node.js?
Node.js provides traditional **callback-based** APIs, which require passing a function to handle results, and **Promise-based** APIs, which return a Promise object. Promises allow chaining with .then()/.catch() or using async/await, resulting in cleaner, more manageable asynchronous code.

Q9: What is a control flow function?
Control flow functions manage the execution order of asynchronous tasks. Libraries like

Async.js offer utilities (series, parallel) to sequence callbacks. With Promises and async/await, developers can write asynchronous code that reads like synchronous code, improving readability and flow control.

Q10: Why is Node.js single-threaded?
Node.js uses a single-threaded event loop to handle concurrency without the overhead of thread management. Blocking operations are delegated to background threads in libuv's thread pool or handled by the OS kernel, combining simplicity with high performance.

Q11: What are the pros and cons of Node.js?
Pros: Non-blocking I/O for high throughput, unified JavaScript stack, large npm ecosystem, easy scalability. Cons: Not ideal for CPU-heavy tasks, potential callback complexity (though async/await helps), variable maturity of some npm modules, and security risks in a fast-moving ecosystem.

Q12: What is the difference between Node.js vs AJAX?
Node.js is a server-side JavaScript runtime for building backend applications. AJAX is a client-side technique using asynchronous HTTP requests from the browser. AJAX fetches or submits data without reloading the page; Node.js processes requests on the server.

Q13: What are the challenges with Node.js?
Challenges include handling CPU-bound tasks due to single-threading, managing complex callback flows (mitigated by promises/async-await), ensuring security with rapidly changing dependencies, and maintaining code quality and best practices in large teams.

Q14: Mention the framework most commonly used in Node.js
Express.js is the leading web framework for Node.js. It offers minimal, unopinionated routing, middleware support, and a simple API for building web and mobile applications. Its performance, flexibility, and extensive ecosystem make it the de facto choice.

Web Services Questions

Q1: What is a Web Service?
A web service is a standardized method for different applications to communicate over a network, typically using protocols like SOAP or REST. It exposes functions or resources through HTTP, exchanging XML or JSON messages to facilitate interoperable, loosely-coupled systems.

Q2: What are the advantages of Web Services?
Web services enable platform- and language-agnostic interoperability, remote execution of business logic, loose coupling, reusability, and easy integration in SOA architectures. They can be orchestrated into complex workflows and scaled independently.

Q3: What are different types of Web Services?
Common types include **SOAP** (XML envelopes, WSDL for contracts), **RESTful** (resource-

based, HTTP verbs, often JSON), **XML-RPC** and **JSON-RPC** (remote procedure calls over HTTP). **GraphQL** is an emerging alternative for client-driven queries.

Q4: What is SOAP?
SOAP (Simple Object Access Protocol) is a messaging protocol using XML envelopes to structure requests and responses. It relies on WSDL for service descriptions and supports standards for security (WS-Security), transactions, and reliable messaging.

Q5: What are advantages of SOAP Web Services?
SOAP offers strict standards, formal contracts via WSDL, built-in fault handling, WS-* extensions (security, transactions), and stateful operations, making it suitable for enterprise-level, mission-critical systems requiring advanced features.

Q6: What are disadvantages of SOAP Web Services?
SOAP is verbose and complex, requiring extensive XML parsing and strict schemas. It demands heavier tooling, can be slower due to message size, and increases development and debugging effort compared to simpler RESTful services.

Q7: What is REST Web Services?
REST (Representational State Transfer) is an architectural style using HTTP methods (GET, POST, PUT, DELETE) on resources identified by URIs. It emphasizes stateless interactions, cacheability, and a uniform interface, often exchanging JSON.

Q8: What are advantages and disadvantages of REST web services?
Advantages: simplicity, performance, scalability, cache support, widespread tooling. Disadvantages: lack of formal contracts, no built-in security standards (relying on HTTP/HTTPS), and less rigorous error handling compared to SOAP.

Q9: Which factors to be considered for website planning? Why is it important? Which design issues?
Consider target audience, functionality, content hierarchy, SEO, accessibility, scalability, security, and budget. Proper planning ensures usability, performance, and ROI. Design issues include poor navigation, inconsistent branding, slow load times, non-responsive layouts, and accessibility violations, all harming user engagement.

EJB Questions

Q1: What is EJB?
EJB (Enterprise JavaBeans) is a Java EE server-side component architecture for encapsulating business logic. The EJB container manages lifecycle, transactions, security, and concurrency, allowing developers to focus on application logic.

Q2: What are the types of Enterprise Bean or EJB?
There are three main types: **Session Beans** (Stateless, Stateful, Singleton), **Message-Driven**

Beans (asynchronous processing), and the deprecated **Entity Beans** (replaced by JPA entities).

Q3: What is session bean?
Session beans encapsulate business operations. They can be stateless (no client state), stateful (maintain client state), or singleton (one shared instance).

Q4: What is stateless session bean?
Stateless session beans do not retain client state between method calls. They're pooled for performance and reused across multiple clients, ideal for scalable, shared services.

Q5: What is stateful session bean?
Stateful session beans maintain conversational state across multiple method calls for a client, useful for multi-step processes like shopping carts.

Q6: What is singleton session bean?
Singleton session beans ensure a single shared instance per application, often used for shared resources, caching, or application-wide initialization.

Q7: What is Entity Bean?
Entity Beans were EJB components representing persistent data, but have been superseded by JPA entities for simpler, more flexible ORM mapping.

Q8: What is Message Driven Bean?
MDBs are asynchronous EJB components that listen to JMS queues or topics, processing messages without direct client calls, enabling decoupled event-driven architectures.

Q9: Explain Lifecycle of EJB.
EJB lifecycles vary: container instantiates the bean, injects resources, invokes business methods, and eventually removes or passivates it (for stateful), with @PostConstruct and @PreDestroy callbacks available.

Q10: When to use Enterprise bean?
Use EJB when you need declarative transactions, security, concurrency management, and remote access for enterprise-grade business logic in Java EE environments.

Q11: What are the benefits of Enterprise beans?
Benefits include container-managed transactions, security, lifecycle and concurrency handling, remote invocation, and scalability. Developers focus on business logic rather than infrastructure.

Spring Questions

Q1: What is Spring?
Spring is an open-source Java framework that simplifies enterprise application

development through Inversion of Control (IoC), dependency injection, aspect-oriented programming, and comprehensive modules for data access, web MVC, security, and more.

Q2: What are the advantages of Spring framework?
Spring promotes loose coupling, modular architecture, easy testing, declarative transaction management, integration with various technologies (JPA, JMS), and lightweight containers, improving developer productivity and application maintainability.

Q3: What are the modules of Spring framework?
Key modules include Core (IoC), AOP, Data Access (JDBC, ORM), MVC, Security, Transaction, Boot, and Integration, covering dependency injection, web, data, messaging, and microservices.

Bootstrap Questions

Q1: Explain what is Bootstrap?
Bootstrap is a popular front-end framework by Twitter for building responsive, mobile-first websites. It provides ready-made CSS and JavaScript components like a grid system, typography, forms, buttons, navbars, and modals.

Q2: Explain why to choose Bootstrap for building websites?
Bootstrap speeds development with its pre-built, consistent design components, ensures cross-browser compatibility, supports responsive layouts out-of-the-box, and offers extensive documentation and community support for customization.

Q3: What are the key components of Bootstrap?
Core components include the 12-column grid system, utility classes (spacing, display), responsive navigation bars, forms, buttons, tables, modals, carousels, and JavaScript plugins (tooltips, dropdowns, collapse).

Struts Questions

Q1: What is Struts?
Apache Struts is an open-source MVC framework for Java web applications. It separates business logic, presentation, and navigation, using Action classes, JSPs with custom tags, and configuration files (struts.xml).

Q2: Explain features of Struts.
Features include MVC architecture, tag libraries for forms and UI, centralized configuration, validation framework, internationalization support, and easy integration with other technologies.

Q3: Explain working of Struts.
A request goes to ActionServlet, which reads struts-config.xml to map to an Action class. The Action executes business logic, returns an ActionForward, and Struts dispatches to the corresponding JSP view.

Q4: Explain Struts tags.
Struts provides custom JSP tags (e.g., <html:form>, <bean:write>, <logic:iterate>) to simplify form handling, data display, and control flow without raw HTML or scriptlets.

Q5: Explain Struts 2 Action class
In Struts 2, Action classes implement com.opensymphony.xwork2.Action or extend ActionSupport. Each method returns a result string (e.g., "success"), which maps to a view or chain of actions in struts.xml or annotations.

Q6: Explain Struts 2 interceptors
Interceptors are reusable pre- and post-processing components (e.g., validation, logging). They wrap Action execution in an interceptor stack, handling cross-cutting concerns declaratively and improving modularity.

Q7: What are the Steps to create Struts 2 web application?

1. Set up project with Struts 2 libraries.
2. Configure struts.xml.
3. Create Action classes.
4. Define result mappings.
5. Develop JSP views using Struts tags.
6. Deploy to a servlet container (e.g., Tomcat).

Ruby on Rails Questions

Q1: List the features of Ruby?
Ruby is a dynamic, object-oriented language with expressive syntax, metaprogramming, garbage collection, mixins via modules, blocks and procs, duck typing, and a rich standard library focused on developer happiness.

Q2: Explain pattern matching operations in Ruby.
Ruby 2.7+ adds pattern matching with the case/in syntax. It destructures objects and arrays, matches values or types against patterns, and extracts data, simplifying complex conditionals and data handling.

Q3: Explain builtin methods for arrays and lists.
Ruby arrays support methods like each, map, select, reject, reduce, push, pop, shift, unshift, and slice, enabling iteration, transformation, filtering, and manipulation of elements.

Q4: List the Uses of Ruby?
Ruby powers web development (Rails), scripting, automation, prototyping, DevOps (Chef, Puppet), data processing, web scraping, and command-line utilities thanks to its readability and extensive gem ecosystem.

Q5: Explain builtin string operations in Ruby
Strings support upcase, downcase, capitalize, split, gsub, sub, strip, concat, interpolation (#{}), length, and pattern matching with regex, offering versatile text processing.

Q6: Example: keyboard input and screen output in Ruby

ruby

CopyEdit

```
print "Enter name: "

name = gets.chomp

puts "Hello, #{name}!"
```

gets reads input; chomp removes newline; puts outputs with a newline; print without newline.

Q7: How a method is created in Ruby? Example

ruby

CopyEdit

```
def greet(name)

  "Hello, #{name}"

end

puts greet("World")
```

Use def and end; parameters are optional; methods return the last evaluated expression.

Q8: How a Rails application is constructed?
rails new app_name scaffolds an MVC app with directories for models, views, controllers, config/, db/, app/assets/, and Gemfile. It follows “convention over configuration” for rapid development.

Q9: Explain form handling with Rails.
Rails uses form helpers (form_with, form_for) tied to models, generating CSRF tokens, appropriate HTTP methods, and parameter mapping automatically, simplifying validation and data submission.

Q10: Explain directory structure for a Rails Application
Key folders: app/ (models, views, controllers), config/ (routes, environment), db/ (migrations, seeds), lib/, public/ (static assets), log/, test/ or spec/, and vendor/.

Q11: How Rails reacts to a request for a static document?
Static files in public/ are served directly by the web server (e.g., Puma, Nginx) without invoking the Rails stack, improving performance by bypassing application code.