U23ITT43 – WEB TECHNOLOGY

Question Bank: 2 Mark Q&A (Units I–V) + 16 Mark Questions

UNIT I – WEBSITE BASICS

PART A – 2 Mark Questions with Answers

1. What is the Internet?

The Internet is a global network of interconnected computers that communicate using standardized protocols (TCP/IP). It enables sharing of information and resources worldwide through services like WWW, email, FTP, and more.

2. What is a Web Protocol?

A Web Protocol is a set of rules governing data exchange over the web. Common protocols include HTTP (HyperText Transfer Protocol) for web pages, HTTPS (secure HTTP), FTP for file transfer, and SMTP for email.

3. Define URL.

URL (Uniform Resource Locator) is the address used to access resources on the internet. It consists of: Protocol (http://), Domain name (www.example.com), Path (/page), and optional Query string (?key=value).

4. What is a Domain Name?

A Domain Name is a human-readable address (e.g., www.google.com) that maps to an IP address via DNS (Domain Name System). It consists of a hostname, domain, and top-level domain (TLD) like .com, .org, .in.

5. What is a Web Browser?

A Web Browser is a software application used to access and display web pages. It sends HTTP requests to web servers and renders the received HTML/CSS/JS. Examples: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.

6. What is a Web Server?

A Web Server is software/hardware that hosts websites and responds to client HTTP requests by serving HTML pages and other resources. Examples: Apache HTTP Server, Nginx, Microsoft IIS. It listens on port 80 (HTTP) or 443 (HTTPS).

7. What is the working principle of a website?

When a user types a URL, the browser sends an HTTP request to the web server. The server processes the request and responds with HTML/CSS/JS files. The browser parses and renders these files to display the webpage to the user.

8. What is client-side scripting?

Client-side scripting refers to scripts (like JavaScript) that execute in the user's web browser rather than on the server. They handle dynamic content, form validation, animations, and interactive UI without server round-trips.

9. What is server-side scripting?

Server-side scripting involves scripts (PHP, JSP, Node.js) that run on the web server to generate dynamic HTML content before sending it to the browser. It handles database operations, authentication, and business logic.

10. What is the difference between static and dynamic websites?

Static websites display fixed content (HTML files) that doesn't change unless manually edited. Dynamic websites generate content dynamically based on user input, database data, or server-side processing. Dynamic sites use server-side scripting like PHP or Node.js.

PART B – 16 Mark Questions

1. Explain the fundamental concepts of computer networks as applied to the web. Describe web protocols (HTTP, HTTPS, FTP, SMTP), URL structure, and Domain Name System (DNS) with neat diagrams.

2. Explain the working principle of a website in detail. Describe the roles of web browsers and web servers, the HTTP request-response cycle, and how web pages are rendered.

3. Describe the process of creating a website. Explain client-side scripting and server-side scripting with suitable examples and their differences.

4. Explain Internet Overview including its history, architecture, services, and how web browsers communicate with web servers using HTTP protocol.

5. Compare and contrast static and dynamic websites. Explain the technologies used for each and describe the role of URL, DNS, and domain names in web communication.

UNIT II – WEB DESIGNING

PART A – 2 Mark Questions with Answers

1. What is HTML?

HTML (HyperText Markup Language) is the standard language for creating web pages. It uses tags (like <h1>, <p>,) to structure content. HTML5 is the latest version with new semantic elements and multimedia support.

2. What are HTML Form Elements?

HTML Form elements collect user input. Key elements: <form>, <input> (text, password, radio, checkbox, submit), <select> (dropdown), <textarea>, <button>, and <label>. Forms submit data to a server using GET or POST methods.

3. What is CSS?

CSS (Cascading Style Sheets) is a style sheet language used to control the visual presentation of HTML elements. It defines properties like color, font, margin, padding, and layout. CSS can be inline, internal (style tag), or external (.css file).

4. What are CSS Selectors?

CSS Selectors target HTML elements to apply styles. Types: Element selector (p), Class selector (.classname), ID selector (#idname), Attribute selector ([type='text']), Pseudo-class (:hover), and Pseudo-element (::before). They control specificity and cascading.

5. What is the CSS Box Model?

The CSS Box Model describes the rectangular boxes generated for elements. It consists of: Content (actual content), Padding (space inside border), Border (surrounds padding), and Margin (space outside border). Total width = content + padding + border + margin.

6. What are CSS Backgrounds and Borders?

CSS Background properties: background-color, background-image, background-repeat, background-position, background-size. CSS Border properties: border-style (solid, dashed), border-width, border-color, border-radius (for rounded corners).

7. What are CSS Text Effects?

CSS Text Effects control text appearance: text-shadow (adds shadow), text-overflow (ellipsis for overflow), word-wrap, text-transform (uppercase/lowercase), letter-spacing, line-height, and word-spacing. CSS3 added text-stroke and gradient text effects.

8. What are CSS Animations?

CSS Animations allow elements to transition between styles over time. Key properties: @keyframes (defines animation steps), animation-name, animation-duration, animation-timing-function (ease, linear), animation-iteration-count, and animation-direction.

9. What is CSS Multiple Column Layout?

CSS Multi-column Layout divides content into multiple columns like a newspaper. Properties: column-count (number of columns), column-width, column-gap (space between columns), column-rule (line between columns), and column-span (element spans all columns).

10. What is User Interface (UI) in web design?

User Interface (UI) in web design refers to the visual elements and interactive components users interact with: buttons, forms, menus, icons. Good UI design follows principles of consistency, feedback, simplicity, and accessibility for better user experience.

PART B – 16 Mark Questions

1. Explain HTML Form Elements and Input types in detail. Describe all input types (text, email, number, date, file, radio, checkbox) with examples and form validation attributes.

2. Describe CSS3 Selectors, Box Model, Backgrounds and Borders in detail with examples. Explain how the box model affects element sizing and layout.

3. Explain CSS Text Effects and CSS Animations in detail. Describe @keyframes, animation properties, transitions, and transform effects with examples.

4. Describe CSS Multiple Column Layout and Media Elements in HTML. Explain responsive design using CSS media queries with examples.

5. Explain the concept of User Interface design in web development. Describe CSS Flexbox and Grid layout systems and how they improve web page design.

UNIT III – CLIENT-SIDE PROCESSING AND SCRIPTING

PART A – 2 Mark Questions with Answers

1. What is JavaScript?

JavaScript is a lightweight, interpreted, client-side scripting language used to make web pages interactive. It runs in the browser, handles DOM manipulation, event handling, form validation, and asynchronous operations (AJAX). It is also used server-side via Node.js.

2. What are Variables and Data Types in JavaScript?

JavaScript variables are declared with var, let, or const. Data types: Primitive types (Number, String, Boolean, null, undefined, Symbol, BigInt) and Reference types (Object, Array, Function). JavaScript is dynamically typed — type is determined at runtime.

3. What are JavaScript Statements and Operators?

JavaScript statements are instructions executed by the browser (if, for, while, switch). Operators: Arithmetic (+,-,*,/,%), Assignment (=,+=), Comparison (==,===,!=,!==,>,<), Logical (&&,||,!), Ternary (?:), and typeof operator.

4. What are Literals and Functions in JavaScript?

Literals are fixed values in code: numeric (42), string ('hello'), boolean (true/false), array ([1,2,3]), object ({key:value}). Functions are reusable code blocks: function declaration, function expression, arrow functions (()=>{}), and immediately invoked function expressions (IIFE).

5. What are JavaScript Objects?

JavaScript objects are collections of key-value pairs. Created using: object literals ({name:'John', age:25}), constructor functions (new Object()), or ES6 classes. Built-in objects: Math, Date, JSON, RegExp. Objects support properties and methods.

6. What are JavaScript Arrays?

JavaScript arrays store ordered collections of values. Key methods: push(), pop(), shift(), unshift(), splice(), slice(), map(), filter(), reduce(), forEach(), find(), sort(), and join(). Arrays are zero-indexed and can hold mixed data types.

7. What are Built-in Objects in JavaScript?

Built-in JavaScript objects: Math (Math.round, Math.random, Math.floor), Date (new Date(), getDate(), getMonth()), String (toUpperCase, split, indexOf), Array (map, filter, reduce), JSON (JSON.parse, JSON.stringify), and RegExp for pattern matching.

8. What is Regular Expression in JavaScript?

A Regular Expression (RegExp) is a pattern used to match character combinations in strings. Syntax: /pattern/flags. Common flags: g (global), i (case-insensitive), m (multiline). Methods: test(), match(), replace(), search(). Used for form validation.

9. What is Exception Handling in JavaScript?

Exception handling manages runtime errors gracefully. Syntax: try { risky code } catch(error) { handle error } finally { always runs }. The throw statement creates custom errors. Error types: TypeError, ReferenceError, SyntaxError, RangeError.

10. What is Event Handling in JavaScript?

Event handling responds to user actions or browser events. Events: click, mouseover, keydown, submit, load, change. Methods: addEventListener('event', function), removeEventListener(). Event object provides details. Event propagation: bubbling and capturing.

PART B – 16 Mark Questions

1. Explain JavaScript Variables, Data Types, Statements, and Operators in detail with examples. Describe var, let, and const and their scope differences.

2. Describe JavaScript Functions (declaration, expression, arrow functions) and Objects in detail. Explain prototypes and the 'this' keyword with examples.

3. Explain JavaScript Arrays and Built-in Objects (Math, Date, JSON) with detailed examples of their methods and practical use cases.

4. Describe Regular Expressions in JavaScript. Explain patterns, flags, and methods (test, match, replace) with examples. Also explain form validation using RegExp.

5. Explain Exception Handling and Event Handling in JavaScript. Describe try-catch-finally, custom errors, DOM events, event listeners, and event propagation (bubbling and capturing).

UNIT IV – TYPESCRIPT

PART A – 2 Mark Questions with Answers

1. What is TypeScript?

TypeScript is a strongly-typed superset of JavaScript developed by Microsoft. It adds optional static typing, interfaces, generics, and modern ES6+ features. TypeScript code is transpiled to JavaScript using the TypeScript compiler (tsc). It improves code quality and IDE support.

2. What are TypeScript Basics?

TypeScript basics include: Type annotations (let name: string), Type inference (compiler infers types), Interfaces (define object shapes), Enums (named constants), Classes with access modifiers, Generics, and Decorators. TypeScript files use .ts extension.

3. What are Data Types in TypeScript?

TypeScript data types: Primitive (string, number, boolean, null, undefined, symbol), Array (number[]), Tuple ([string, number]), Enum (enum Color {Red, Blue}), Any (disables type checking), Unknown, Void (function returns nothing), Never (never returns), and Union types (string | number).

4. What is Destructuring in TypeScript?

Destructuring extracts values from arrays or object properties into distinct variables. Array destructuring: const [a, b] = [1, 2]. Object destructuring: const {name, age} = person. Used with function parameters, default values, rest patterns, and nested structures.

5. What is Spread Syntax in TypeScript?

Spread syntax (...) expands an iterable (array/object) into individual elements. Array spread: [...arr1, ...arr2] merges arrays. Object spread: {...obj1, ...obj2} merges objects. Used for copying arrays/objects, passing multiple

arguments, and merging collections.

6. What are Interfaces in TypeScript?

Interfaces define the structure (shape) of an object in TypeScript. They specify property names and types without implementation. Syntax: interface Person { name: string; age: number; }. Interfaces support optional properties (?), readonly, method signatures, and extending other interfaces.

7. What are Generics in TypeScript?

Generics allow writing reusable, type-safe code that works with multiple types. Syntax: function identity<T>(arg: T): T { return arg; }. Generic constraints: <T extends string>. Generic interfaces, classes, and utility types (Partial<T>, Required<T>, Pick<T,K>, Omit<T,K>).

8. What are Modules in TypeScript?

TypeScript modules organize code into separate files with their own scope. Export: export const x = 1; or export default class. Import: import { x } from './file'; or import * as mod from './file'. Modules prevent global namespace pollution and enable code reuse.

9. What are Name Spaces in TypeScript?

Namespaces (formerly 'internal modules') organize code within a single file or across files. Syntax: namespace MyApp { export interface User {} export class Service {} }. They prevent name collisions. Accessed as MyApp.User. Modules are preferred over namespaces in modern TypeScript.

10. What are Loops and Collections in TypeScript?

TypeScript supports: for loop, while loop, do-while loop, for...of (iterates values), for...in (iterates keys). Collections: Array<T>, Map<K,V> (key-value pairs), Set<T> (unique values), WeakMap, WeakSet. TypeScript provides full type safety for all collection operations.

PART B – 16 Mark Questions

1. Explain TypeScript Basics and its Data Types in detail. Compare TypeScript with JavaScript. Describe type annotations, type inference, and TypeScript compilation process.

2. Describe Destructuring and Spread syntax in TypeScript with detailed examples. Explain how they work with arrays, objects, and function parameters.

3. Explain Interfaces and Generics in TypeScript in detail. Describe interface extension, optional properties, generic constraints, and generic utility types with examples.

4. Describe working with Classes in TypeScript. Explain access modifiers (public, private, protected), abstract classes, inheritance, and decorators with examples.

5. Explain Modules, Namespaces, Functions, Loops, and Collections in TypeScript. Describe ES6 module system, ambient modules, and TypeScript collection types (Array, Map, Set).

UNIT V – SERVLETS AND DATABASE CONNECTIVITY

PART A – 2 Mark Questions with Answers

1. What is a Java Servlet?

A Java Servlet is a server-side Java program that handles HTTP requests and generates dynamic web content (HTML). Servlets run inside a servlet container (like Apache Tomcat). They extend HttpServlet and override doGet() and doPost() methods.

2. What is Servlet Architecture?

Servlet Architecture: Client sends HTTP request → Web Server receives it → Servlet Container loads and instantiates Servlet → Calls service() → Calls doGet()/doPost() → Servlet generates response → Container sends HTTP response back to client.

3. What is the Servlet Life Cycle?

Servlet Life Cycle: (1) Loading – servlet class loaded by container, (2) Instantiation – object created, (3) init() – called once for initialization, (4) service() – called for each request, dispatches to doGet/doPost, (5) destroy() – called once when servlet is unloaded.

4. What is the difference between GET and POST?

GET appends form data to URL (visible, limited length, cached, bookmarkable). Used for non-sensitive data retrieval. POST sends data in request body (hidden, unlimited length, not cached). Used for sensitive data (passwords), file uploads, and form submissions that change server state.

5. What are Sessions in Servlets?

Sessions maintain state between HTTP requests (HTTP is stateless). HttpSession object stores user data across requests. Methods: request.getSession(), session.setAttribute(), session.getAttribute(), session.invalidate(). Session is identified by JSESSIONID cookie or URL rewriting.

6. What are Cookies in Servlets?

Cookies are small text files stored in the browser to maintain state. In Servlets: Create: new Cookie('name','value'); response.addCookie(cookie). Read: request.getCookies(). Set expiry: cookie.setMaxAge(seconds). Cookies store session IDs, preferences, and login status.

7. What is JDBC?

JDBC (Java Database Connectivity) is an API that enables Java programs to interact with relational databases. Steps: (1) Load driver (Class.forName), (2) Create connection (DriverManager.getConnection), (3) Create Statement, (4) Execute query, (5) Process ResultSet, (6) Close connection.

8. What is Database Connectivity in web applications?

Database connectivity links web applications to databases (MySQL, Oracle, PostgreSQL). JDBC provides the bridge between Java servlets and databases. Connection string format: jdbc:mysql://localhost:3306/dbname. Used for CRUD operations: Create, Read, Update, Delete.

9. What is a Simple Interactive Web Application?

A Simple Interactive Web Application uses servlets to process user input from HTML forms, perform database operations via JDBC, and return dynamic HTML responses. Example: Login system — HTML form → Servlet → JDBC query → MySQL → Servlet generates response page.

10. What are Database Applications in Servlets?

Database Applications using Servlets integrate frontend HTML forms, backend Servlet logic, and database storage. Common applications: Student registration, Login authentication, Product catalog, Shopping cart. Uses MVC pattern: JSP (View), Servlet (Controller), JavaBeans/JDBC (Model).

PART B – 16 Mark Questions

1. Explain Java Servlet Architecture and Servlet Life Cycle in detail. Describe the roles of Web Server, Servlet Container, and the stages of a servlet (init, service, destroy) with a neat diagram.

2. Describe Form GET and POST actions in Servlets. Explain how to handle HTML form submissions, process request parameters, and generate dynamic HTML responses.

3. Explain Sessions and Cookies in Servlets. Describe HttpSession management, cookie creation/reading, URL rewriting, and how state is maintained in web applications.

4. Describe JDBC (Java Database Connectivity) in detail. Explain the steps to connect to a MySQL database, execute queries, handle ResultSets, and perform CRUD operations from a Servlet.

5. Design and explain a complete simple interactive database web application using Servlets and JDBC. Show the flow from HTML form submission through Servlet processing to database operation and response generation.