MAA OMWATI DEGREE COLLEGE HASSANPUR

(PALWAL)

NOTES

BCA 2nd SEM

WEB DEVELOPMENT-II

Unit-1

📘 XML (eXtensible Markup Language)

🔹 Introduction

- XML stands for **eXtensible Markup Language**.
- Developed by W3C to **store and transport data**.
- Not a replacement for HTML but complements it.
- **Self-descriptive**: data is stored along with meaningful tags.

Example:

xml

CopyEdit

```
<student>
   <name>Jane Doe</name>
   <age>21</age>
</student>
```

🔹 Syntax Rules

- Tags are case-sensitive.
- Must have a **root element**.
- Elements must be **properly nested**.
- All tags must be **closed**.
- Attributes must be quoted.

Example:

xml

CopyEdit

```
<book id="b101">
  <title>XML Basics</title>
</book>
```

- **Document Structure**

An XML document has:

1. **Prolog**
2. **Root element**
3. **Nested child elements**

Example:

xml

CopyEdit

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <book>
    <title>XML Fundamentals</title>
  </book>
</library>
```

- **Document Type Definitions (DTD)**
  - Defines **structure** and **rules** for XML document.
  - Can be **internal** or **external**.

Internal DTD:

xml

CopyEdit

```
<!DOCTYPE note [
  <!ELEMENT note (to, from, heading, body)>
  <!ELEMENT to (#PCDATA)>
]>
```

External DTD:

xml

CopyEdit

```
<!DOCTYPE note SYSTEM "note.dtd">
```

- **Namespaces**

Used to **avoid element name conflicts** when combining XML documents.

xml

CopyEdit

```
<book xmlns:math="http://www.math.org">
   <math:equation>x + y = z</math:equation>
</book>
```

- **XML Schema (XSD)**
  - XML Schema defines the structure using **XML-based syntax**.
  - More powerful than DTD.

Example:

xml

CopyEdit

```
<xs:element name="price" type="xs:decimal"/>
```

- **Displaying XML Documents**

1. Raw XML Display

Browsers display XML as a **tree view**.

2. Using CSS

xml

CopyEdit

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

css

CopyEdit

```
book {
   color: blue;
}
```

3. Using XSLT (eXtensible Stylesheet Language Transformations)

- Used to transform XML into HTML.

Example:

xml

CopyEdit

```
<xsl:template match="/book">
  <html>
```

```
  <body>
    <h1><xsl:value-of select="title"/></h1>
  </body>
 </html>
</xsl:template>
```

🔹 XML Processors

These are tools that parse XML documents:

- **DOM Parser**: Loads whole XML tree into memory.
- **SAX Parser**: Event-based, reads sequentially.
- Examples: Apache Xerces, MSXML

🔹 Web Services with XML

- **SOAP** (Simple Object Access Protocol) uses XML for structured messaging.
- XML is used for:
  - Data serialization
  - Service responses and requests
  - Configuration files

💻 ECMAScript (JavaScript)

🔹 Introduction

ECMAScript is the standardized specification for **JavaScript**. The versions are denoted as ES1, ES5, ES6, ESNext, etc.

🔹 **ECMAScript Versions**

- **ES3 (1999)**: Added regular expressions.
- **ES5 (2009)**: Strict mode, JSON support.
- **ES6 (2015)**: Major update introducing classes, modules, arrow functions, etc.
- ES7+ (ES2016 and onwards): Added features like includes(), async/await, etc.

🔹 **Key ES5 Features**

- **Strict Mode**: Enforces stricter parsing.

javascript

CopyEdit

"use strict";

- **Array Methods**: forEach, map, filter, reduce
- **Object.create()**
- **JSON.parse() / JSON.stringify()**

🌟 **ES6 (ECMAScript 2015+) and Beyond**

🔹 **Var Declarations and Hoisting**

javascript

CopyEdit

console.log(x); // undefined

var x = 10;

- var is function-scoped.
- Hoisted to the top of the function.

🔹 **Let Declaration**

- Block-scoped.
- Not hoisted.

javascript

CopyEdit

```
let y = 20;
```

🔹 **Constant Declaration**

- Immutable reference.

javascript

CopyEdit

```
const PI = 3.14;
```

🔹 **Function with Default Parameter Values**

javascript

CopyEdit

```
function greet(name = "Guest") {
   console.log("Hello, " + name);
}
```

🔹 **Default Parameter Expressions**

javascript

CopyEdit

```
function area(radius, pi = Math.PI) {
   return pi * radius * radius;
}
```

- **Unnamed Parameters (Rest Parameters)**

javascript

CopyEdit

```
function sum(...numbers) {
   return numbers.reduce((a, b) => a + b);
}
```

- **Spread Operator**

javascript

CopyEdit

```
let arr = [1, 2];
let newArr = [...arr, 3, 4];
```

- **Arrow Functions**

javascript

CopyEdit

```
const square = x => x * x;
```

- No this binding

- Compact syntax

🔹 **Object Destructuring**

javascript

CopyEdit

```
const person = { name: "John", age: 25 };
const { name, age } = person;
```

🔹 **Array Destructuring**

javascript

CopyEdit

```
const [a, b] = [10, 20];
```

🔹 **Sets and Maps**

- **Set**: Unique values.

javascript

```
const set = new Set([1, 2, 2, 3]); // {1, 2, 3}
```

- **Map**: Key-value pairs with any data type.

javascript

```
const map = new Map();
map.set('a', 100);
```

🔹 **Array.find() and Array.findIndex()**

javascript

```
arr.find(x => x > 10);
arr.findIndex(x => x > 10);
```

🔹 Template Strings

```
javascript
let name = "Alice";
let msg = `Hello, ${name}!`;
```

🔹 JavaScript Classes

```
javascript
class Person {
    constructor(name) {
        this.name = name;
    }
    greet() {
        console.log("Hi " + this.name);
    }
}
```

🔹 Callbacks

```
javascript
function fetchData(callback) {
    setTimeout(() => callback("Data fetched"), 1000);
}
```

- **Promises**

javascript

CopyEdit

```
let promise = new Promise((resolve, reject) => {
    resolve("Done");
});

promise.then(result => console.log(result));
```

- **Async/Await**

javascript

CopyEdit

```
async function fetchData() {
    let result = await fetch("https://api.com/data");
    let data = await result.json();
    console.log(data);
}
```

Unit-2

🌐 AJAX (Asynchronous JavaScript and XML)

🔷 Introduction to AJAX

AJAX stands for **Asynchronous JavaScript and XML**. It is not a programming language but rather a **technique** that uses various web technologies to create asynchronous web applications.

- Originally used to fetch XML, but now widely used with **JSON**.
- It allows web pages to **communicate with the server without reloading the page**.
- Developed to improve user experience and create **faster, more dynamic web pages**.

🔷 Need for AJAX

Before AJAX, web applications used the **synchronous model**, where:

1. A request was sent to the server.
2. The server processed it.
3. The entire web page was reloaded with the response.

This was **slow and inefficient**. AJAX solves this by:

- Updating parts of a web page, without refreshing the entire page.
- Improving performance and interactivity.
- Reducing server load and network usage.

Examples of AJAX in use:

- Live search suggestions (e.g., Google search)
- Chat applications
- Form validation
- Auto-updating feeds (e.g., Twitter, Instagram)

🔷 How AJAX Works

AJAX uses the following components:

- **HTML** – For structure.
- **CSS** – For styling.
- **JavaScript** – For dynamic behavior.
- **XMLHttpRequest / Fetch API** – For sending/receiving data.
- **Server-side script** – PHP, Node.js, ASP.NET, etc., to process data.
- **JSON / XML / HTML** – For data interchange.

🔷 Working of AJAX

🔁 Steps:

1. User action triggers a JavaScript function (e.g., clicking a button).
2. JavaScript creates an **XMLHttpRequest** object.
3. Request is sent to the server **asynchronously**.
4. Server processes the request and sends back a **response**.
5. JavaScript receives the response and updates the **DOM** (HTML content) without reloading the page.

javascript

CopyEdit

```
let xhr = new XMLHttpRequest();
xhr.open("GET", "data.json", true);
xhr.onload = function() {
  if (xhr.status == 200) {
    document.getElementById("output").innerHTML = xhr.responseText;
  }
};
xhr.send();
```

🔷 Handling AJAX Requests and Responses

AJAX can be implemented using:

- **XMLHttpRequest** (older method)
- **Fetch API** (modern method)

Example using Fetch:

javascript

CopyEdit

```
fetch('data.json')
  .then(response => response.json())
  .then(data => console.log(data));
```

🗂️ Data Formats Used in AJAX

✅ 1. XML

- Extensible Markup Language.
- Structured but **verbose**.
- Once popular, now mostly replaced by JSON.

✅ **2. JSON**

- **JavaScript Object Notation**
- Lightweight, easy to parse.
- Native to JavaScript.
- Looks like JavaScript object literals.

Example:

json

CopyEdit

```
{
  "name": "Alice",
  "age": 25
}
```

🔃 **Working with JSON Data**

When receiving JSON from the server:

✅ **Parse JSON:**

javascript

CopyEdit

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

✅ **Convert to JSON:**

javascript

CopyEdit

```
let data = { name: "Bob" };
let json = JSON.stringify(data);
```

📄 **Loading Data with AJAX**

🔷 **Loading HTML with AJAX**

javascript

CopyEdit

```
$("#result").load("page.html");
```

🔷 **Loading XML with AJAX**

javascript

CopyEdit

```
$.ajax({
  type: "GET",
  url: "data.xml",
  dataType: "xml",
  success: function(xml) {
    $(xml).find('item').each(function() {
      $("#result").append($(this).text());
    });
  }
});
```

🔷 Loading JSON with AJAX

javascript

CopyEdit

```
$.getJSON("data.json", function(data) {
  $("#output").html(data.name);
});
```

⚙️ jQuery

🔷 Introduction to jQuery

- A fast, small, and feature-rich **JavaScript library**.
- Simplifies **HTML DOM manipulation**, **event handling**, **AJAX**, and **animations**.
- Compatible with all major browsers.

🔷 Basic jQuery Example

html

CopyEdit

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
  $(document).ready(function() {
    $("button").click(function() {
      $("#demo").text("Hello jQuery!");
    });
```

```
});
</script>
```

🔷 Need for jQuery

- Cross-browser compatibility.
- Less code, more functionality.
- Easier to handle events, AJAX, and animations.
- Abstracts complex native JavaScript syntax.

🔍 jQuery Element Handling

✅ Finding and Selecting Elements

javascript

CopyEdit

```
$("#id")       // ID selector
$(".class")    // Class selector
$("p")         // Tag selector
```

✅ Getting Element Content

javascript

CopyEdit

```
let text = $("#demo").text();     // Get text content
let html = $("#demo").html();     // Get HTML content
```

✅ Updating and Changing Content

javascript

CopyEdit

```
$("#demo").text("New text");
$("#demo").html("<b>Bold Text</b>");
```

✅ Inserting Elements / New Content

javascript

CopyEdit

```
$("#demo").append("<p>Appended</p>");
$("#demo").prepend("<p>Prepended</p>");
```

✅ Getting and Setting Attributes

javascript

CopyEdit

```
let href = $("a").attr("href");      // Get attribute
$("a").attr("href", "https://new.com"); // Set attribute
```

✅ Getting and Setting CSS

javascript

CopyEdit

```
$("#box").css("color", "red");       // Set CSS
let color = $("#box").css("color");  // Get CSS
```

✅ Using each() Method

javascript

CopyEdit

```
$("li").each(function(index) {
  console.log(index + ": " + $(this).text());
});
```

🧠 Events in jQuery

javascript

CopyEdit

```
$("#btn").click(function() {
  alert("Clicked!");
});
```

✅ Event Object

javascript

CopyEdit

```
$("button").click(function(event) {
  console.log(event.type); // click
});
```

💫 Effects and Animation

✅ Basic Effects

- .show()

- .hide()
- .toggle()
- .fadeIn(), .fadeOut()
- .slideDown(), .slideUp()

javascript

CopyEdit

```
$("#demo").fadeIn();
```

✅ Animating CSS Properties

javascript

CopyEdit

```
$("#box").animate({
  left: "200px",
  opacity: 0.5
}, 1000);
```

🔍 DOM Traversing in jQuery

- .parent(), .children(), .siblings(), .find(), .closest()

javascript

CopyEdit

```
$("p").parent().css("border", "1px solid red");
```

📜 Working with Forms

javascript

CopyEdit

```
let name = $("#name").val();     // Get input value
$("#name").val("John");          // Set input value
```

- .submit(), .change(), .focus(), .blur() – for form events

📚 JavaScript Libraries, jQuery & AJAX

- jQuery provides a simplified interface for handling **AJAX**:

javascript

CopyEdit

```
$.ajax({
  url: "data.json",
  method: "GET",
  success: function(data) {
    $("#output").html(data.name);
  }
});
```

- Libraries like Axios, Fetch API, or jQuery can be used depending on project needs.
- jQuery is still widely used in legacy systems, but newer frameworks like **React**, **Vue**, and **Angular** offer more structure.

Unit-3

🔹 Introduction to Web Servers

A **web server** is both:

1. **Software** – that handles HTTP requests and serves HTTP responses (typically HTML, CSS, JS, etc.).
2. **Hardware** – a physical machine or virtual server that stores web resources and delivers them over the internet.

Key Functions:

- Handle incoming HTTP requests from clients (like browsers).
- Serve content like web pages, images, videos.
- Execute server-side code if needed (like PHP, Node.js).

Examples: Apache, Nginx, Microsoft IIS, Node.js (with HTTP module or Express), Lighttpd.

🔹 HTTP Transactions

HTTP (HyperText Transfer Protocol) is a stateless protocol used for client-server communication over the web.

🔁 An HTTP transaction typically includes:

1. **Client Request**:
    - Sent by a browser or client using GET, POST, PUT, DELETE, etc.
    - Includes headers, optional body (in POST/PUT), cookies, etc.
2. **Server Response**:

- Includes a status code (200 OK, 404 Not Found), headers, and body (HTML, JSON, etc.).

Example HTTP Request:

vbnet

CopyEdit

GET /index.html HTTP/1.1

Host: example.com

Example HTTP Response:

bash

CopyEdit

HTTP/1.1 200 OK

Content-Type: text/html

<body>Welcome</body>

- **Multitier Application Architecture**

A multitier (or multilayered) architecture separates concerns into distinct layers:

1. **Client Tier (Presentation)** – UI/UX layer (HTML, CSS, JS).
2. **Web Tier (Application Logic)** – Business logic, routes, controller logic.
3. **Data Tier (Database)** – Stores and retrieves data (SQL or NoSQL DB).

Benefits:

- Scalability
- Separation of concerns

- Easier maintenance and testing

🔹 Client-Side vs Server-Side Scripting

Feature	Client-Side	Server-Side
Runs on	Browser	Server
Languages	HTML, CSS, JavaScript	PHP, Node.js, Python, Ruby
Speed	Faster (local execution)	Depends on server processing
Security	Less secure	More secure (can access DB, files)
Usage	Interactivity, UI changes	Data processing, authentication

---

💻 SERVER-SIDE SCRIPTING WITH NODE.JS

🔹 Getting to Know Node.js

- Created by **Ryan Dahl** in 2009.
- Built on **Chrome's V8 JavaScript Engine**.
- Makes JavaScript usable on the **server side**.
- Embraces **asynchronous, event-driven** programming.

🔹 How Node.js Changed JavaScript Forever

- Traditionally, JavaScript was **only client-side**.
- Node.js enabled **full-stack development** using one language—JavaScript.

- It popularized **non-blocking I/O** and real-time applications like chat, gaming, etc.

🔹 Features of Node.js

- Asynchronous & Non-blocking
- Single-threaded but event-driven
- Fast execution via V8
- Huge npm ecosystem
- Cross-platform
- Scalable for I/O-heavy tasks

🔹 When to Use / Not Use Node.js

✅ **Use Node.js:**

- Real-time apps (chat, gaming)
- REST APIs and microservices
- Streaming services
- SPAs with dynamic content

❌ **Avoid for:**

- CPU-intensive apps (image processing)
- Complex multi-threaded computations

🔹 Asynchronous Callbacks

Node.js uses **callbacks and promises** for async tasks.

Example:

javascript

CopyEdit

```
fs.readFile("file.txt", "utf-8", (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

- **The NoSQL Movement**
  - Traditional databases (SQL) use rigid schemas.
  - **NoSQL** databases like MongoDB are **schema-less**, flexible, and scalable.
  - Ideal for storing JSON-like data and unstructured content.

- **Node and MongoDB in the Wild**
  - Together, they form the **ME(N) stack** (MongoDB, Express, Node).
  - Used in apps like Netflix, Uber, Trello, LinkedIn, etc.

- **Hello World in Node**

javascript

CopyEdit

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.end('Hello World!');
});
```

server.listen(3000);

- **package.json**
  - Metadata file for any Node project.
  - Defines dependencies, scripts, version, and configuration.

json

CopyEdit

```
{
  "name": "my-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

- **Modules in Node.js**
  - Node supports **CommonJS modules** (require()).
  - Code can be modularized into separate files.

javascript

CopyEdit

```
// math.js
exports.add = (a, b) => a + b;
```

- **Built-in Modules**

- http, fs, path, url, events, os, crypto, etc.

Example:

javascript

CopyEdit

```
const fs = require('fs');
fs.readFile("file.txt", "utf-8", (err, data) => console.log(data));
```

🔹 HTTP Module

Used to build basic servers.

javascript

CopyEdit

```
const http = require("http");
http.createServer((req, res) => {
  res.write("Hello!");
  res.end();
}).listen(3000);
```

🔹 Events Module

javascript

CopyEdit

```
const EventEmitter = require('events');
const emitter = new EventEmitter();
emitter.on('greet', () => console.log("Hello Event!"));
emitter.emit('greet');
```

🔹 **Node Package Manager (npm)**

- World's largest software registry.
- Install dependencies: npm install express
- Scripts and package management.

🚀 **Node.js with Express**

🔹 **Express.js**

- Minimalist web framework for Node.
- Provides routing, middleware, and more.

Install:

bash

CopyEdit

npm install express

🔹 **Middleware**

- Functions that execute between the request and response.

javascript

CopyEdit

```
app.use((req, res, next) => {
  console.log("Request Received");
  next();
});
```

- **Routing in Express**

javascript

CopyEdit

```
app.get('/', (req, res) => {
  res.send("Home Page");
});
app.post('/submit', (req, res) => {
  res.send("Form submitted");
});
```

- **CRUD Operations in Express**

javascript

CopyEdit

```
app.get("/items", ...);    // Read
app.post("/items", ...);   // Create
app.put("/items/:id", ...);// Update
app.delete("/items/:id",...);// Delete
```

- **Web Server using Express**

javascript

CopyEdit

```
const express = require("express");
const app = express();
```

```
app.get("/", (req, res) => res.send("Welcome"));
app.listen(3000);
```

🔹 Making it Live on Heroku

1. Sign up at https://heroku.com
2. Install Heroku CLI
3. Initialize Git & add remote:

```
bash
CopyEdit
heroku create
git push heroku main
```

🗄️ Node.js with MongoDB

🔹 Basics of MongoDB

- NoSQL database
- Stores data as JSON-like documents (BSON)
- Collections and Documents instead of Tables and Rows

🔹 MongoDB CRUD Operations

```
javascript
CopyEdit
// Insert
db.users.insertOne({ name: "Alice" });
```

```
// Find
db.users.find({ name: "Alice" });

// Update
db.users.updateOne({ name: "Alice" }, { $set: { age: 30 } });

// Delete
db.users.deleteOne({ name: "Alice" });
```

- **Building Data Model with Mongoose**
  - Mongoose is an ODM for MongoDB + Node.js.
  - Helps define schemas, validations, relationships.

javascript

CopyEdit

```
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  name: String,
  age: Number
});
const User = mongoose.model("User", userSchema);
```

- **Defining Simple Mongoose Schemas**

javascript

CopyEdit

```
const postSchema = new mongoose.Schema({
  title: { type: String, required: true },
  body: String,
  createdAt: { type: Date, default: Date.now }
});
```

- **Build Node Express App with MongoDB**

javascript

CopyEdit

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();

mongoose.connect("mongodb://localhost:27017/test");

const User = mongoose.model("User", new mongoose.Schema({ name: String }));
app.get("/", async (req, res) => {
  const users = await User.find();
  res.send(users);
});

app.listen(3000);
```

Unit-4

🔹 Basic Knowledge of Websites

A **website** is a collection of related web pages hosted on a web server and accessible via the Internet using a web browser. Websites fall into two main categories:

1. **Static Websites**: Content is fixed and written directly in HTML/CSS.
2. **Dynamic Websites**: Content is generated dynamically using server-side scripting like PHP, ASP.NET, etc., often based on user interaction or database queries.

🔹 Introduction to Dynamic Website

Dynamic websites are capable of interacting with databases, managing sessions, and changing content based on user behavior or inputs. Technologies like PHP, Python, Ruby, Java, and ASP.NET power dynamic websites.

Features:

- Content changes dynamically (news, profiles, shopping carts).
- User interaction (forms, logins).
- Backend processing (database read/write).
- Examples: Facebook, YouTube, Amazon.

🔹 Introduction to PHP

PHP (Hypertext Preprocessor) is a powerful, server-side scripting language designed for web development. It is embedded in HTML and is used to create dynamic web pages.

- Originally created by **Rasmus Lerdorf** in 1994.
- PHP scripts are executed on the server, and the output is sent to the browser.
- The file extension for PHP files is .php.

Characteristics:

- Open-source
- Interpreted language
- Cross-platform (works on Windows, Linux, macOS)
- Loosely typed

🔹 Why PHP and Its Scope

✅ Why PHP?

- Easy to learn for beginners.
- Widely supported by hosting providers.
- Vast ecosystem with libraries, frameworks (Laravel, CodeIgniter), and CMS (WordPress, Joomla).
- Integrates well with databases like MySQL.
- Large community and documentation.

📈 Scope of PHP

- Full-stack development (when combined with JavaScript and MySQL).
- Backend APIs for mobile apps.

- Web-based enterprise software.
- Content Management Systems.
- E-commerce applications.

🔹 XAMPP and WAMP Installation

To run PHP locally, you need a **web server + PHP interpreter + database server**.

✔ **XAMPP:**

- Cross-platform
- Includes Apache, MariaDB, PHP, and Perl

✔ **WAMP:**

- Windows-only
- Includes Apache, MySQL, PHP

Installation Steps:

1. Download from apachefriends.org (for XAMPP).
2. Install and run the **Apache** and **MySQL** services.
3. Place your .php files in the htdocs folder.
4. Run PHP files via http://localhost/filename.php.

🔧 PHP Programming Basics

🔹 Syntax of PHP

php

CopyEdit

```
<?php
echo "Hello World!";
?>
```

Important Notes:

- PHP code is enclosed in <?php ?> tags.
- Statements end with a semicolon ;.
- Comments:

```
php
CopyEdit
// Single line comment
/* Multi-line comment */
```

- **Embedding PHP in HTML**

```
html
CopyEdit
<html>
 <body>
  <h1>Welcome</h1>
  <?php echo "This is a dynamic message."; ?>
 </body>
</html>
```

- **Embedding HTML in PHP**

```
php
```

CopyEdit

```
<?php
echo "<h2>Hello World!</h2>";
?>
```

- **Introduction to PHP Variables**
  - Start with $ sign: $name = "Alice";
  - Variables are case-sensitive.
  - No need to declare data type explicitly.

- **Understanding PHP Data Types**
  1. String ("Hello")
  2. Integer (42)
  3. Float (3.14)
  4. Boolean (true, false)
  5. Array (array(1,2,3))
  6. Object
  7. NULL

- **Using Operators**
  1. **Arithmetic**: + - * / %
  2. **Assignment**: = += -=
  3. **Comparison**: == != < > ===
  4. **Logical**: && || !

5. **Increment/Decrement**: ++ --

🔹 **Using Conditional Statements**

✅ **if, else if, else:**

```
php
CopyEdit
$score = 85;
if ($score > 90) {
    echo "Excellent";
} elseif ($score > 75) {
    echo "Good";
} else {
    echo "Try again";
}
```

✅ **switch Statement:**

```
php
CopyEdit
$day = "Tuesday";
switch ($day) {
  case "Monday":
    echo "Start of the week";
    break;
  case "Tuesday":
    echo "Second day";
```

```
    break;
  default:
    echo "Unknown day";
}
```

🔹 **Using Loops**

✅ **while Loop:**

```
php
CopyEdit
$i = 1;
while ($i <= 5) {
    echo $i;
    $i++;
}
```

✅ **for Loop:**

```
php
CopyEdit
for ($i = 0; $i < 5; $i++) {
    echo $i;
}
```

🧠 **PHP Functions**

🔹 Creating User-Defined Functions

php

CopyEdit

```
function greet($name) {
   return "Hello, $name!";
}
echo greet("Alice");
```

🔹 Using Predefined PHP Functions

Examples:

- strlen("hello") → returns 5
- strtoupper("hello") → "HELLO"
- date("Y-m-d") → current date

🧮 Arrays in PHP

🔹 Creating Arrays

php

CopyEdit

```
$colors = array("Red", "Green", "Blue");
```

🔹 Modifying Array Elements

php

CopyEdit

```
$colors[0] = "Yellow";
```

- **Processing Arrays with Loops**

```
php
CopyEdit
foreach ($colors as $color) {
    echo $color . "<br>";
}
```

- **Grouping Form Selections with Arrays**

```
html
CopyEdit
<select name="fruits[]" multiple>
  <option>Apple</option>
  <option>Banana</option>
</select>
php
$selected = $_POST['fruits'];
foreach ($selected as $fruit) {
    echo $fruit;
}
```

- **Using Array Functions**
  - count($arr)
  - sort($arr)
  - array_push($arr, "item")
  - in_array("item", $arr)