UNIT - 4

Front-End Web UI Frameworks and Tools

Introduction to Bootstrap

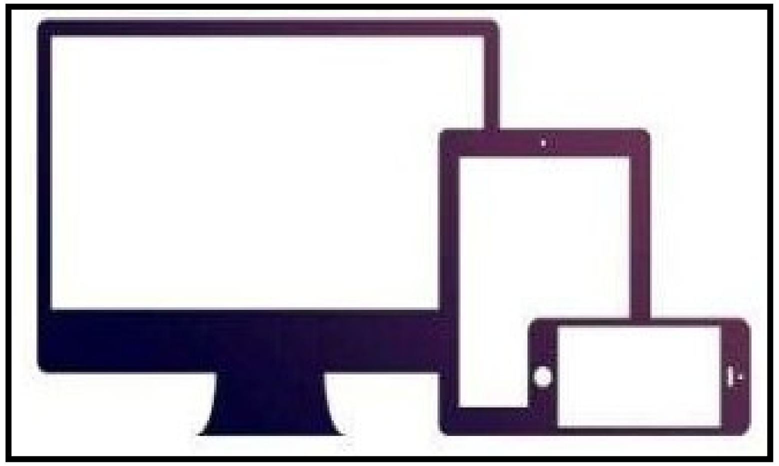
Bootstrap is a **popular front-end framework** used for designing **responsive and mobile-first websites** quickly and efficiently. It includes **pre-styled CSS and JavaScript components**, such as buttons, forms, grids, and navigation bars, which help developers create professional-looking websites without writing much custom CSS.

What is Bootstrap?

- Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- It is absolutely free to download and use.
- It is a front-end framework used for easier and faster web development.
- It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals and many others.
- It can also use JavaScript plug-ins.
- It facilitates you to create responsive designs.
-

Contd.

- Bootstrap is an **open-source CSS framework** developed by Twitter to make web development easier. It provides:
  - **A responsive grid system**
  - **Pre-designed UI components**
  - **Powerful JavaScript plugins**
  - **Customizable themes**



- Bootstrap follows a **mobile-first approach**, meaning it is designed to work **seamlessly on all screen sizes** (mobile, tablet, and desktop).

Why Use Bootstrap?

- Responsive Design – Automatically adapts to different screen sizes.
- Pre-Styled Components – Ready-to-use UI elements (buttons, cards, forms, etc.).
- Faster Development – Saves time by using predefined styles and layouts.
- Grid System – Uses a flexible 12-column grid for easy layout design.
- Cross-Browser Compatibility – Works smoothly on all modern browsers.
- Consistent Design – Ensures a professional look across all devices.

What is a responsive website

- A website is called responsive website which can automatically adjust itself to look good on all devices, from smart phones to desktops etc.

Bootstrap Versions

- **Bootstrap 1 (2011)**
  - Initial release
  - Basic grid system
  - Simple UI components
- **Bootstrap 2 (2012)**
  - Responsive design introduced
  - Fluid grid system
- **Bootstrap 3 (2013)**
  - Mobile-first approach
  - Flat design

Contd.

- **Bootstrap 4 (2018)**
  - Flexbox-based grid
  - Improved components
  - Dropped support for IE9
- **Bootstrap 5 (2021)**
  - Removed jQuery dependency
  - Improved grid and utilities
  - Better CSS customization

Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

You can:

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN (Content Delivery Network).

How to Use Bootstrap?

There are two ways to integrate Bootstrap into your project:

1.Using Bootstrap CDN (Easy & Quick Method)

•Just add the following links in your HTML file:

```
<!-- Bootstrap CSS -->

<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.mi
n.css" rel="stylesheet">
```

Contd.

```
<!-- Bootstrap JS (For interactive components) -->

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bund
le.min.js"></script>
```

2. Installing via npm (For Developers)

```
npm install bootstrap
```

Then import it into your project:

```
import 'bootstrap/dist/css/bootstrap.min.css';

import 'bootstrap/dist/js/bootstrap.bundle.min.js';
```

Full-Stack Web Development

- Full-stack web development involves designing, developing, and managing both the **frontend (client-side)** and **backend (server-side)** of a web application.
- The **front end** consists of the **user interface** (or UI), and the **back end** handles the **business logic** and application workflows that run behind the scenes.
- A **Full-Stack Web Developer** is proficient in multiple technologies required to build an entire web application.

Components of Full-Stack Web Development

1. Frontend Development (Client-Side)
2. Backend Development (Server-Side)
3. Database Management
4. DevOps & Deployment

Contd.

1. Frontend Development (Client-Side)

This is the part of the application that users interact with directly.

Technologies Used:

- HTML – Creates the structure of web pages
- CSS – Defines styling and layout (Bootstrap, Tailwind CSS)
- JavaScript – Adds interactivity and dynamic content
- Frontend Frameworks/Libraries: React.js, Angular, Vue.js

Contd.

2. Backend Development (Server-Side)

The backend is responsible for handling business logic, database operations, and server communication.

Technologies Used:

Programming Languages: Node.js, Python (Django, Flask), Java (Spring Boot), PHP, Ruby on Rails

Backend Frameworks: Express.js, FastAPI

APIs: RESTful APIs, GraphQL for client-server communication

Contd.

3. Database Management

Databases store and manage application data.

Types of Databases:

SQL Databases – MySQL, PostgreSQL

NoSQL Databases – MongoDB, Firebase

Contd.

4. DevOps & Deployment

A full-stack developer should also be familiar with hosting and deploying applications.

Tools Used:

Version Control: Git, GitHub, GitLab

Cloud Hosting Services: AWS, Heroku, Netlify, Vercel

Containerization: Docker, Kubernetes

Popular Stacks

- **LAMP stack:** JavaScript - Linux - Apache - MySQL - PHP
- **LEMP stack:** JavaScript - Linux - Nginx - MySQL - PHP
- **MEAN stack:** JavaScript - MongoDB - Express - AngularJS - Node.js
- **MERN stack:** JavaScript – MongoDB – Express - React.js - Node.js
- **Django stack:** JavaScript - Python - Django - MySQL
- **Ruby on Rails:** JavaScript - Ruby - SQLite - Rails

Version Control System (VCS)

A **Version Control System (VCS)** is a tool that helps developers manage changes to code over time. It keeps track of modifications, enables collaboration, and allows rollback to previous versions.

Types of Version Control Systems:

1. Local Version Control (LVC)

2. Centralized Version Control (CVCS)

3. Distributed Version Control (DVCS) (Most Popular)

Git

- Git is a distributed version control system designed for tracking changes in source code during software development.
- Developed by Linus Torvalds in 2005, initially to manage the Linux kernel development.
- Used to work collaboratively over a team , track code changes.

Git vs Github

- Git: The version control system itself, installed locally on a developer's machine.
- GitHub: A web-based platform built around Git, providing a hosting service for Git repositories and additional collaboration features.
- Git manages the versioning of code, while GitHub adds features like issue tracking, pull requests, project management tools, and team collaboration.

Setting Up Git

Step 1: Install Git

- Download Git from git-scm.com.
- Follow the installation instructions for your OS (Windows, macOS, Linux).
- After installation , Configure Git
- Verify installation using:

```
git --version
```

Contd.

Step 2: Configure Git

Before using Git, set up your username and email:

```
git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"
```

Check your configurations:

```
git config --list
```

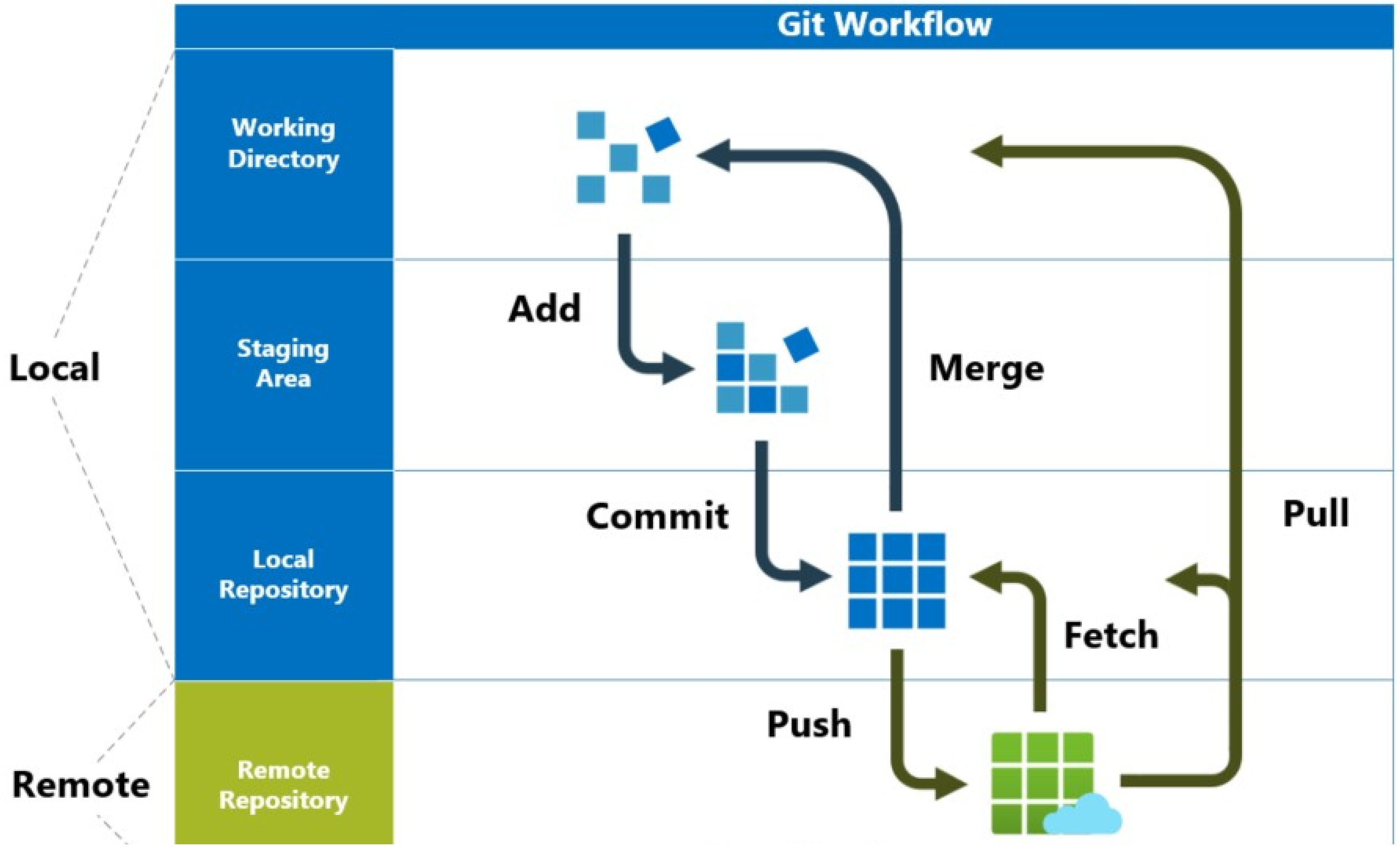
Basic Git Commands

- git init: - Initializes a new Git repository in the current directory.
- git add: - Adds changes in the working directory to the staging area, preparing them for the next commit.
- git commit: - Records changes to the repository, creating a snapshot of the staged changes.
- git push: - Uploads local repository content to a remote repository, usually hosted on a service like GitHub.
- git pull: - Fetches changes from a remote repository and merges them into the local repository.

Contd.

- git clone: - Creates a local copy of a remote repository.
- git branch: -Lists, creates, or deletes branches within the repository.
- git merge: - Combines changes from one branch into another.
- git checkout: -Switches branches or restores working tree files to a specific state.
- git status: - Displays the current state of the working directory and the staging area

Git Workflow



Contd.

1. Initializing a Repository

This creates a new Git repository in the current directory.

```
git init
```

2. Checking Status

Shows the current state of your working directory.

```
git status
```

Contd.

3. Adding Files to Staging

```
git add <filename>  # Add a specific file
git add .           # Add all modified files
```

4. Committing Changes

Commits staged files with a message.

```
git commit -m "Initial commit"
```

Contd.

5. Check if a remote is set

```
git remote –v
```

6. Add a Remote Repository

If you haven't added a remote repository, you need to link your local repository to a GitHub/GitLab repository. Replace <your-repo-url> with your actual repository URL from GitHub:

```
git remote add origin <your-repo-url>
```

Contd.

7. Verify the Remote

```
git remote -v
```

You should see something like:

```
origin  https://github.com/yourusername/your-repo.git (fetch)
origin  https://github.com/yourusername/your-repo.git (push)
```

8. Push the Code

Now you can push your changes:

```
git push -u origin main
```

Contd.

5. Viewing Commit History

```
git log
```

6. Creating and Switching Branches

```
git branch feature-branch  # Create a new branch
git switch feature-branch  # Switch to the new branch
```

Contd.

7. Merging Branches

```
git checkout main  # Switch to the main branch
git merge feature-branch  # Merge changes from feature-branch into main
```

Git Hub

GitHub Account

Go to GitHub and sign up for an account

Create a Repository on GitHub

- Now that you have made a GitHub account, sign in, and create a new Repo
- Public (if you want the repo to be viewable for anyone) or
- Private (if you want to choose who should be able to view the repo).
- Either way, you will be able to choose who can **contribute** to the repo.

Contd.

- Then click "Create repository".
- Since we have already set up a local Git repo, we are going to push that to GitHub.
- *git remote add origin URL* specifies that you are adding a remote repository, with the specified URL, as an origin to your local Git repo. Now we are going to push our master branch to the origin url, and set it as the default remote branch.

Online Git Repositories

1. GitHub (github.com)
2. GitLab (gitlab.com)
3. Bitbucket (bitbucket.org)
4. SourceForge (sourceforge.net)
5. AWS CodeCommit (aws.amazon.com/codecommit)
6. Azure DevOps Repos (azure.microsoft.com)
7. Google Cloud Source Repositories (cloud.google.com/source-repositories)

Contd.

Creating a GitHub Repository

1.Go to GitHub and sign in.

2.Click New Repository.

3.Give it a name and click Create Repository.

Connecting Local Repo to GitHub

1. Initialize Git (if not done already)

```
git init
```

Contd.

2. Add a Remote Repository

```
git remote add origin https://github.com/your-username/repository-name.git
```

3. Push Code to GitHub

```
git push -u origin main
```

4. Cloning a GitHub Repository

```
git clone https://github.com/your-username/repository-name.git
```

Contd.

5. Pulling the Latest Changes

```
git pull origin main
```

6. Pushing Changes to GitHub

```
git push origin main
```

Difference Between Git and GitHub

Feature	**Git**	**GitHub**
Definition	A **distributed version control system (DVCS)** that tracks code changes.	A **cloud-based platform** that hosts Git repositories.
Purpose	Manages local and remote code repositories.	Provides a web-based UI for Git, collaboration tools, and hosting.
Hosting	Git itself **does not provide hosting**—it runs locally on your machine.	GitHub hosts Git repositories on **cloud servers**.
Collaboration	Git allows local development and **push/pull from remote repositories**.	GitHub enables **team collaboration, pull requests, issue tracking, and CI/CD**.
Access	**Command-line based** (git init, git commit, git push).	**Web-based UI** with visual tools + CLI support (GitHub Desktop).

Node.js

- Node.js is an open source, cross platform runtime environment for server side and networking application.
- Basically, Node.js is server side programming language like PHP, C#, Python, Java etc.
- Node.js is cross platform. So it runs well on Linux systems, Mac, can also run on Windows systems.
- It's an runtime environment used for executing JavaScript code without browser..
- We often use node to build back-end services like API's .

Contd.

- Node.js is a JavaScript runtime built on the V8 JavaScript engine. It allows you to run JavaScript on the server side, enabling the development and high performance web application. Node.js comes with a package manager called npm (Node Package Manager) that allows you to easily install and manage third-party libraries.

Key Features of Node.js

- **Asynchronous & Event-Driven** – Uses a non-blocking I/O model for better efficiency.
- **Fast Execution** – Runs on the V8 JavaScript engine (same as Chrome).
- **Single-Threaded but Highly Scalable** – Uses an event loop to handle multiple requests efficiently.
- **Built-in Package Manager (NPM)** – Comes with **Node Package Manager (NPM)** to manage dependencies.
- **Cross-Platform** – Works on Windows, macOS, and Linux.
- **Used for Full-Stack Development** – Works with frameworks like **Express.js, Next.js, and Nest.js** for backend development.

How Node.js Works (Step-by-Step)

- **Client Request** → A user sends a request (e.g., accessing a website or API).
- **Event Loop** → Instead of creating multiple threads, Node.js handles requests using a **single-threaded event loop**.
- **Non-Blocking I/O** → If a request requires data from a database or file system, Node.js does **not** wait for it; it moves to the next request.
- **Callback Execution** → Once data is available, a callback function executes to send the response.
- **Response Sent** → The server sends the response back to the client.

Node.js Installation

Step 1: Download Node.js

Go to the official Node.js website:

•https://nodejs.org/

You will see two versions:

1.LTS (**Long-Term Support**) – Recommended for most users (stable & reliable).

2.Current **(Latest Features**) – Has the newest updates (may be unstable).

- Choose the LTS version for better stability.

Contd.

Step 2: Install Node.js

Step 3: Verify Installation

After installation, open the terminal (or Command Prompt on Windows) and run:

```
node -v
```

To check NPM (Node Package Manager):

```
npm -v
```

First Node.js Program

After installation, let’s test Node.js by writing a simple program.

1. **Create a file named app.js:**

```
console.log("Hello, Node.js!");
```

2. Run the script in the terminal:

```
node app.js
```

Output:

```
Hello, Node.js!
```

Why Use Node.js?

- **Fast & Efficient** – Uses Google’s V8 engine for fast execution.
- **Asynchronous & Non-Blocking** – Handles multiple requests without waiting.
- **Single Language (JavaScript)** – Use JS for both frontend & backend.
- **Scalable** – Suitable for high-traffic applications.
- **Real-Time Apps** – Ideal for chat apps, gaming, live streaming.
- **Cross-Platform** – Runs on Windows, macOS, Linux, Cloud.
- **NPM Ecosystem** – Over 1 million packages available.
- **Works Well with NoSQL (MongoDB)** – Uses JSON for easy integration.
- **Used by Big Companies** – Netflix, Uber, PayPal, LinkedIn use Node.js.

Building a Basic Web Server with Node.js:

```
const http = require('http');
const hostname = 'localhost';
const port = 3001;
const server = http.createServer((req, res) => {
res.statusCode = 200;
res.setHeader('Content-Type', 'text/plain');
res.end('Hello, World!');
});
 server.listen(port, hostname, () => {
console.log(Server is running on http://${hostname}:${port}/);
});
```

NPM (Node Package Manager)

- NPM (**Node Package Manager**) is a tool that helps developers install, manage, and share **JavaScript libraries (called packages or modules**) for Node.js applications.
- It comes **automatically installed with Node.js.**
- It has over **1 million free packages** to speed up development.
- You can **install, update**, and **remove** packages easily.

Why Use NPM?

Feature	Benefit
Package Management	Install and manage third-party libraries easily
Dependency Handling	Keeps track of required modules in package.json
Largest JS Library	Over 1M+ open-source packages available
Speeds Up	Development No need to write everything from scratch
Custom Scripts	Automate tasks using NPM commands

Basic NPM Commands

Command	Description
npm -v	Check installed NPM version
npm init	Create a new package.json file
npm install <package>	Install a package locally
npm install -g <package>	Install a package globally
npm uninstall <package>	Remove a package
npm update <package>	Update a package
npm list	Show installed packages