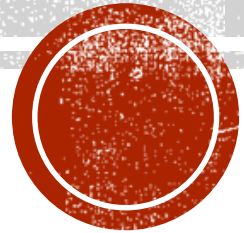
JQUERY



WHY JQUERY

- jQuery is a fast, small, and feature-rich JavaScript library. It simplifies things like HTML document traversal and manipulation, event handling, and animation. Here's why jQuery became popular:
- **Cross-Browser Compatibility**: jQuery handles many of the cross-browser issues that developers face, ensuring that code works consistently across all major browsers.
- **Simplified Syntax**: jQuery provides a simpler, more intuitive syntax for complex tasks, reducing the amount of code you need to write.
- **Rich Set of Plugins**: There are thousands of plugins available for jQuery, which can extend its functionality significantly.
- **AJAX Support**: jQuery simplifies the process of making asynchronous HTTP requests (AJAX) to load data from a server without refreshing the page.

JQUERY FUNDAMENTALS

- **Include the jQuery Library**
- There are two main ways to include the jQuery library in your HTML file:
- **Using a CDN (Content Delivery Network)**: This is the easiest method. You link to a remote copy of the jQuery library hosted by a CDN.
- **Downloading and Hosting Locally**: You download the jQuery library and host it on your own server.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>jQuery Example</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
    <!-- Your content goes here -->
    <script>
        $(document).ready(function() {
            // Your jQuery code goes here
        });
    </script>
</body>
</html>
```

BASIC JQUERY SYNTAX:

- **$()**: The main function for selecting elements.
- **$(document).ready()**: Ensures the DOM is fully loaded before running jQuery code.
- **Example**:

Javascript

```
$(document).ready(function() {
    $("button").click(function() {
        alert("Button clicked!");
    });
});
```

- **Manipulating DOM Elements**
- jQuery provides various methods to manipulate DOM elements. Here are some common methods:

1.html():

- **Description**: Get or set the HTML content of an element.
- **Example**:

Javascript

```
$("#myDiv").html("<p>New content</p>");
```

2.text():

- **Description**: Get or set the text content of an element.
- **Example**:

```javascript
$("#myDiv").text("New text content");
```

3.css():

- **Description**: Get or set CSS properties.
- **Example**:

```javascript
$("#myDiv").css("color", "blue");
```

4.attr():

- **Description**: Get or set attributes.
- **Example**:

```javascript
$("#myLink").attr("href", "https://www.example.com");
```

5.addClass() / .removeClass():

- **Description**: Add or remove CSS classes.
- **Example**:

Javascript

```
$("#myDiv").addClass("highlight");
```

6.hide() / .show():

- **Description**: Hide or show elements.
- **Example**:

Javascript

```
$("#myDiv").hide();
$("#myDiv").show();
```

7.append() / .prepend():

- **Description**: Insert content inside an element.
- **Example**:

```javascript
$("#myDiv").append("<p>Appended content</p>");
```

OVERVIEW OF POPULAR FRAMEWORKS/LIBRARY (REACT, ANGULAR, VUE.JS)

- **React**
- **Description**: A JavaScript library for building user interfaces, developed by Facebook.
- **Key Features**: Component-based architecture, virtual DOM, declarative syntax.
- **Use Cases**: Single-page applications (SPAs), dynamic web apps.
- **Example**: Facebook, Instagram.

Step 1: Install and npm

React requires and npm (Node Package Manager) to manage packages and dependencies. If you haven't installed them yet, follow these steps:

1. **Download and Install** :
   - Go to the .
   - Download and install the LTS version of Node.js, which includes npm.
2. **Verify Installation**:
   - Open a terminal or command prompt.
   - Run the following commands to check if and npm are installed correctly:
   - Node -v

- **Step 2: Create a New React Application**
- The easiest way to create a new React application is by using Create React App, a command-line tool that sets up a new React project with a sensible default configuration.
- **Open a terminal or command prompt.**
- **Navigate to the directory** where you want to create your React project.
- **Run the following command** to create a new React application:

```
npx create-react-app my-react-app
```

- **Replace** my-react-app with the name of your project.
- npx is a package runner tool that comes with npm 5.2+

- **Navigate into the project directory**:
- **Start the development server**:
  - Npm start

•This command starts a local development server and opens your new React application in your default web browser.

•The default URL is http://localhost:3000.

- Step 3: Understand the Project Structure

Your new React project has a basic structure:

```
my-react-app/
├── node_modules/
├── public/
│   └── index.html
├── src/
│   ├── App.css
│   ├── App.js
│   ├── App.test.js
│   ├── index.css
│   ├── index.js
│   ├── logo.svg
│   └── reportWebVitals.js
├── .gitignore
├── package.json
├── README.md
└── yarn.lock
```

- **public/**: Contains the index.html file where your React app is mounted.
- **src/**: Contains your React components and styles.
- : The main component of your application.
- : The entry point of your React application.

Step 4: Edit and Use React Components

React components are the building blocks of a React application.

Let's edit the default App.js to display a simple welcome message.

src/App.js:

src/App.js:

Javascript

```javascript
import React from 'react';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <h1>Welcome to My React App!</h1>
        <p>This is a simple React application.</p>
      </header>
    </div>
  );
}

export default App;
```

Step 5: Add Styles

You can style your React components using CSS. Edit the `App.css` file to add s
custom styles.

src/App.css:

Css

```css
.App {
  text-align: center;
}

.App-header {
  background-color: #282c34;
  padding: 20px;
  color: white;
}
```

Javascript

```javascript
// A simple React component
import React from 'react';
import ReactDOM from 'react-dom';

function App() {
    return <h1>Hello, React!</h1>;
}

ReactDOM.render(<App />, document.getElementById('root'));
```

ANGULAR

- **Description**: A TypeScript-based open-source web application framework, developed by Google.
- **Key Features**: Two-way data binding, dependency injection, component-based structure.
- **Use Cases**: Large-scale enterprise applications.
- **Example**: Google Analytics, Microsoft Office 365.
- **Example**:

USES OF ANGULAR

- Angular is used for creating Single Page Applications (SPAs), which are web applications that load a single HTML page and dynamically update the content as the user interacts with the app. Here are some common uses:
- **Enterprise Web Applications**: Suitable for building complex, large-scale applications like CRM systems, financial dashboards, and content management systems.
- **Progressive Web Apps (PWAs)**: Angular provides tools to create web apps that load like regular web pages but offer functionalities similar to native apps.
- **E-commerce Websites**: Used for building interactive shopping experiences with features like product catalogs, shopping carts, and user authentication.
- **Real-time Applications**: Ideal for applications that require real-time data updates, such as chat applications and live streaming platforms.

Installing Angular

To get started with Angular, you'll need to instal₁ and npm (Node Package Manager)

Step-by-Step Installation Guide

1.Install Angular CLI: The Angular CLI (Command Line Interface) is a tool that helps you create and manage Angular projects. Open your terminal or command prompt and run the following command to install Angular CLI globally:

npm install –g @angular/cli

2. **Create a New Angular Project**: Use the Angular CLI to create a new project. In the terminal, navigate to the directory where you want to create your project and run:

ng new my-angular-app

- **Navigate to the Project Directory**: After creating the project, navigate into the project directory:
- **Run the Development Server**: Start the development server to serve your application locally:
- ng serve

  By default, the application will be accessible at

  http://localhost:4200. Open this URL in your web browser to see your new Angular app in action.

Let's create a simple Angular component to display a welcome message.

Step 1: Generate a New Component

1. In your terminal, within the project directory, run:

```sh
ng generate component welcome
```

This command will create a new component called `welcome` with the necessary files (`welcome.component.ts`, `welcome.component.html`, `welcome.component.css`, and `welcome.component.spec.ts`).

Step 2: Edit the Component

1. **Edit the** `welcome.component.html` : Replace the contents of `src/app/welcome/welcome.component.html` with:

Html

```
<div class="welcome">
  <h1>Welcome to Angular</h1>
  <p>This is a simple Angular component.</p>
</div>
```

2. **Style the Component**: Edit the `src/app/welcome/welcome.component.css` some basic styling:

Css

```css
.welcome {
  text-align: center;
  margin-top: 50px;
}

.welcome h1 {
  color: #1976d2;
}

.welcome p {
  color: #333;
  font-size: 18px;
}
```

3. **Update the Main Template**: Open the main application template file `src/app/app.component.html` and add the `<app-welcome></app-welcome>` include the new component:

Html

```
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="assets/angular.png">
</div>
<app-welcome></app-welcome>
```

4. **Run the Application**: Make sure the development server is running (`ng serve`), and navigate to `http://localhost:4200` in your web browser. You should see the welcome message from the new component.

Typescript

```typescript
// A simple Angular component
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `<h1>Hello, Angular!</h1>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent { }
```

VUE.JS

- **Description**: A progressive JavaScript framework for building user interfaces.
- **Key Features**: Reactive data binding, component-based architecture, ease of integration.
- **Use Cases**: SPAs, interactive web interfaces.
- **Example**: Alibaba, Xiaomi.
- **Example**:

Javascript

```javascript
// A simple Vue.js component
import Vue from 'vue';

new Vue({
  el: '#app',
  data: {
    message: 'Hello, Vue.js!'
  }
});
```

CHOOSING THE RIGHT FRAMEWORK FOR PROJECTS

When choosing the right framework for a project, consider the following factors:

1. **Project Requirements**:
    - Evaluate the complexity and requirements of your project. React might be ideal for dynamic UIs, Angular for large-scale applications, and Vue.jsfor ease of integration and simplicity.

2. **Learning Curve**:
    - Consider the learning curve associated with each framework. Vue.jsis often praised for its gentle learning curve, while Angular may require more time to master due to its comprehensive nature.

3. **Community and Support**:
    - Assess the community support and resources available. React and Angular have large communities and extensive documentation, while Vue.jsalso has growing support and resources.
4. **Performance**:
    - Consider the performance needs of your application. React's virtual DOM can offer performance benefits for dynamic applications, while Angular's two-way data binding can be beneficial for enterprise applications.
5. **Scalability**:
    - Evaluate the scalability of the framework for future growth. Angular is known for its robustness in large-scale applications, while React and Vue.jsoffer flexibility and scalability.

Framework/Library	Key Features	Use Cases	Example Applications
React	Component-based, virtual DOM, declarative	SPAs, dynamic web apps	Facebook, Instagram
Angular	Two-way data binding, dependency injection	Large-scale enterprise applications	Google Analytics, Microsoft Office 365
Vue.js	Reactive data binding, component-based	SPAs, interactive web interfaces	Alibaba, Xiaomi