

Optimumpartners

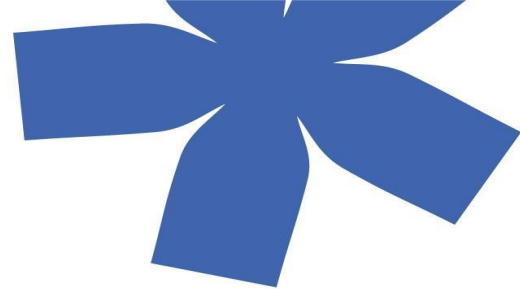
Frontend

**OPTIMUM
PARTNERS**

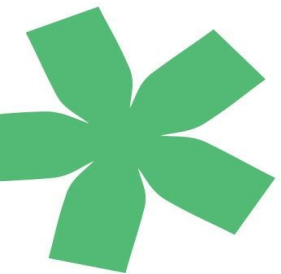


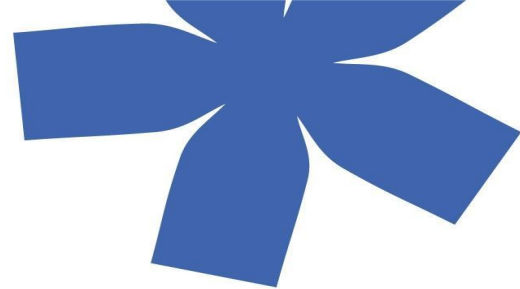
YOUR SUCCESS PARTNER





Essentials of Frontend Development





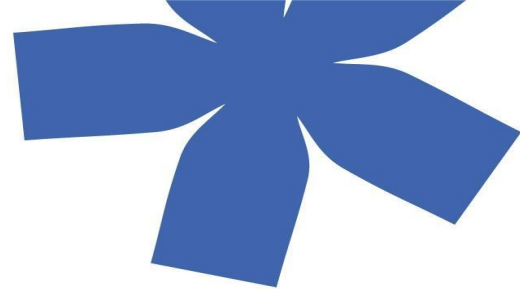
Day 1: HTML Fundamentals - part1

Topics:

- Introduction to HTML
 - HTML Document Structure (<!DOCTYPE>, <html>, <head>, <body>)
 - Basic Tags (<h1> to <h6>, <p>,
, <hr>)
 - Comments in HTML
- Text Formatting & Links
 - Text Formatting Tags (, , <u>, <mark>, <sub>, <sup>)
 - Hyperlinks (<a> tag, href, target, title)
- Lists & Images
 - Ordered Lists (), Unordered Lists (), Definition Lists (<dl>)
 - List Items (), Nested Lists
 - Images (tag, src, alt, width, height)



OPTIMUM PARTNERS



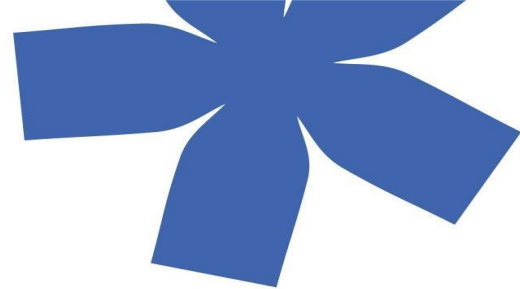
Day 1: HTML Fundamentals - part2

Topics:

- Tables
 - Table Structure (<table>, <tr>, <td>, <th>)
 - Table Attributes (border, colspan, rowspan)
- Forms
 - Form Structure (<form>, <input>, <label>, <button>)
 - Input Types (text, password, email, number, date, radio, checkbox, etc.)
 - Dropdowns (<select>, <option>), Textarea, File Upload
 - Form Attributes (action, method, placeholder, required)



OPTIMUM PARTNERS

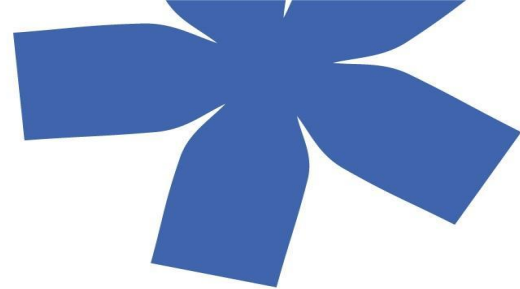


Day 1: HTML Fundamentals - part3

Topics:

- Tables
 - Table Structure (<table>, <tr>, <td>, <th>)
 - Table Attributes (border, colspan, rowspan)
- Forms
 - Form Structure (<form>, <input>, <label>, <button>)
 - Input Types (text, password, email, number, date, radio, checkbox, etc.)
 - Dropdowns (<select>, <option>), Textarea, File Upload
 - Form Attributes (action, method, placeholder, required)





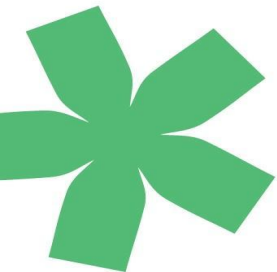
Day 1: HTML Fundamentals - part4

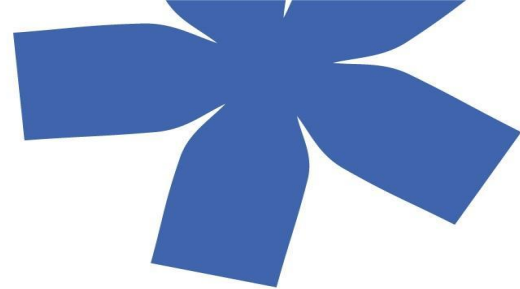
Hands-on:

- Create a simple HTML page with headings, paragraphs, and line breaks
- Build a navigation menu linking to different sections of the same page
- Create a page with different types of lists and images
- Create a timetable or a product comparison table
- Create a registration form with various input types

Exercise:

- Build a basic "About Me" page using headings and paragraphs
- Build a recipe page with ingredients (unordered list) and steps (ordered list)
- Build a table for employee details with headers and merged cells
- Build a survey form with multiple-choice questions and a submit button



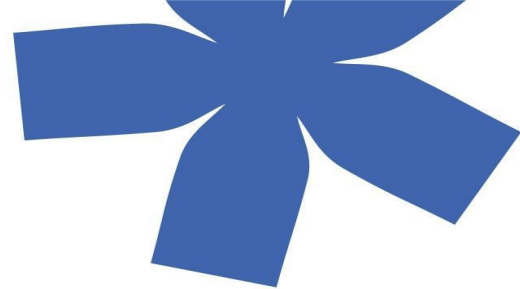


Day 2: CSS Fundamentals - part1

Topics:

- Introduction to CSS
- Inline, Internal, and External CSS
- CSS Syntax (Selectors, Properties, Values)
- Basic Selectors (Element, Class, ID)
- Colors, Backgrounds, & Fonts
- Color Properties (color, background-color)
- Background Properties (background-image, background-repeat, background-position)
- Font Properties (font-family, font-size, font-weight, font-style)
- Box Model
- Box Model Concept (Margin, Border, Padding, Content)
- Box Model Properties (margin, padding, border, width, height)
- Box Sizing (box-sizing: border-box)
- Display & Positioning
- Display Properties (block, inline, inline-block, none)
- Positioning (static, relative, absolute, fixed, sticky)





Day 2: CSS Fundamentals - part2

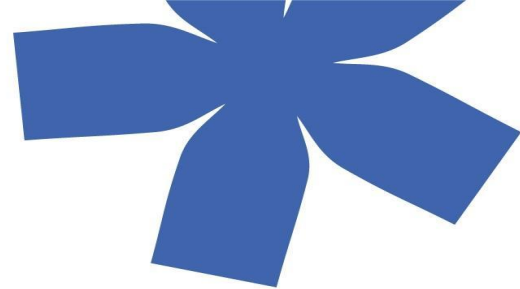
Hands-on:

- Apply inline, internal, and external CSS to a simple HTML page
- Create a colorful page with custom fonts and background images
- Create a layout using the box model properties
- Create a layout with different positioning properties

Exercise:

- Style a basic HTML page using external CSS
- Style a blog page with a background image and custom fonts
- Build a card layout with padding, margin, and borders
- Build a navbar with a dropdown menu using positioning

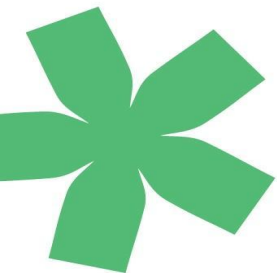


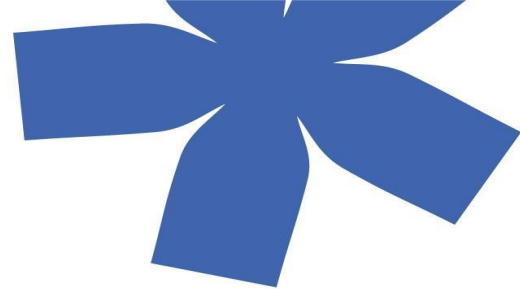


Day 3: CSS Layouts & Responsive Design - part1

Topics:

- Flexbox
- Flexbox Basics (display: flex, flex-direction, justify-content, align-items)
- Flex Properties (flex-grow, flex-shrink, flex-basis)
- CSS Grid
- Grid Basics (display: grid, grid-template-columns, grid-template-rows)
- Grid Properties (gap, grid-column, grid-row)
- Responsive Web Design
- Media Queries (@media)
- Responsive Units (px, em, rem, %, vh, vw)
- Mobile-First Design

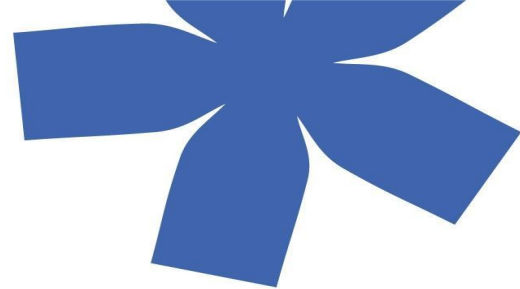




Day 3: CSS Layouts & Responsive Design - part1

- **Hands-on:**
- Create a responsive layout using Flexbox
- Create a complex layout using CSS Grid
- Make a responsive webpage using media queries
- **Exercise:**
- Build a responsive gallery layout using Flexbox
- Build a magazine-style layout using CSS Grid
- Convert a desktop layout to a mobile-friendly layout

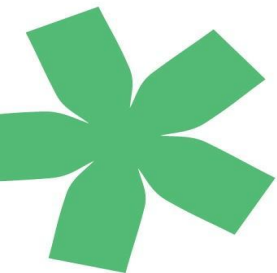




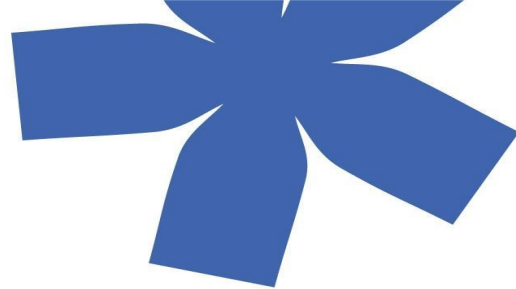
Day 4: Advanced CSS & Preprocessors - part1

Topics:

- Transitions & Animations
- Transitions (transition-property, transition-duration, transition-timing-function)
- Animations (@keyframes, animation-name, animation-duration)
- CSS Variables & Custom Properties
- CSS Variables (--var, var())
- Using Variables for Theming
- CSS Preprocessors (SASS/SCSS)
- Introduction to SASS/SCSS
- Variables, Nesting, Mixins, Partial



OPTIMUM PARTNERS



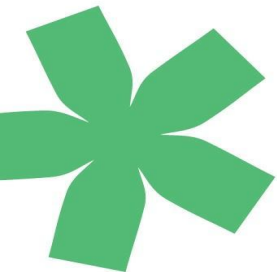
Day 4: Advanced CSS & Preprocessors - part2

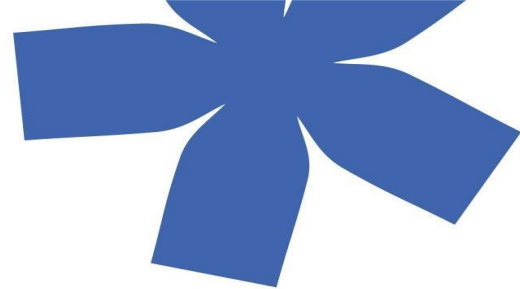
Hands-on:

- Add hover effects and animations to buttons and images
- Create a theme switcher using CSS variables
- Convert a CSS file to SCSS

Exercise:

- Create an animated loading spinner
- Build a dark/light mode toggle for a webpage
- Create a reusable button component using SASS



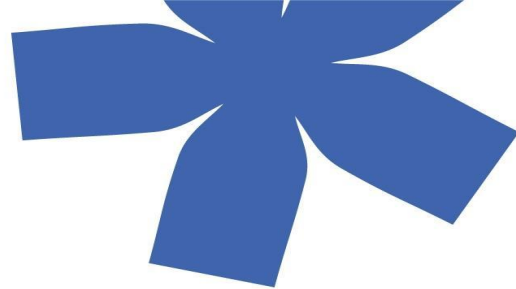


Day 5: CSS Architecture, Accessibility & Cross-Browser Compatibility- part1

Topics:

- CSS Architecture (BEM Methodology)
- BEM (Block, Element, Modifier) Naming Convention
- Organizing CSS for Large Projects
- Accessibility in HTML & CSS
- Semantic HTML for Accessibility
- ARIA Roles and Attributes
- Accessibility Best Practices
- Cross-Browser Compatibility
- Browser-Specific Issues
- Vendor Prefixes
- Tools for Testing Compatibility





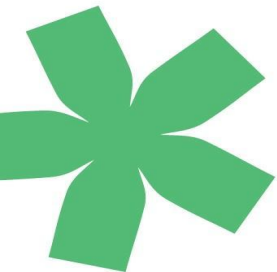
Day 5: CSS Architecture, Accessibility & Cross-Browser Compatibility- part2

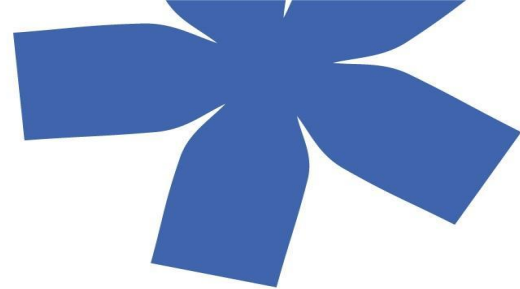
Hands-on:

- Refactor a CSS file using BEM methodology
- Improve the accessibility of an existing webpage
- Test and fix a webpage for cross-browser compatibility

Exercise:

- Apply BEM to a small project
- Build an accessible form with proper labels and ARIA attributes
- Ensure a webpage works consistently across different browsers





Day 6: Introduction to JavaScript & Basics

Topics:

- What is JavaScript?
- JavaScript History & Versions (ES5, ES6+)
- Setting Up the Environment (Browser Console, Node.js, VS Code)
- Writing Your First JavaScript Program
- JavaScript Syntax & Structure
- Variables: var, let, const
- Data Types: Strings, Numbers, Booleans, null, undefined, Symbol, BigInt

Hands-on:

Write a simple "Hello, World!" program

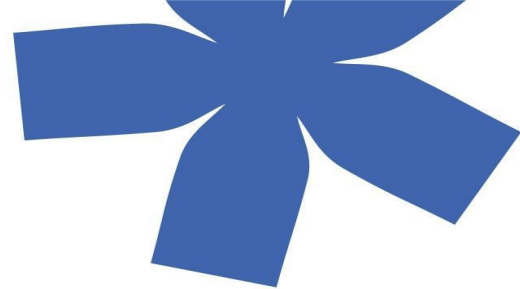
Declare variables and perform basic operations

Exercise:

Write a program to display your name and age using console.log()



OPTIMUM PARTNERS



Day 7: Operators, Control Flow, and Loops

Topics:

- Type Coercion & Type Conversion
- Operators: Arithmetic, Comparison, Logical, Assignment, Ternary
- Conditional Statements: if, else, else if, switch
- Loops: for, while, do-while
- break and continue statements

Hands-on:

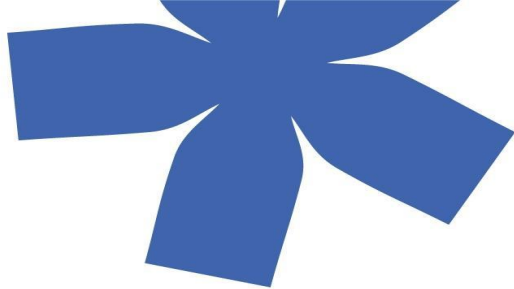
Use comparison and logical operators in conditions
Write a program to check if a number is even or odd
Use loops to print numbers from 1 to 10

Exercise:

Write a program to find the factorial of a number



OPTIMUM PARTNERS



Day 8: Functions & Scope

Topics:

- Function Declaration vs Function Expression
- Arrow Functions (ES6)
- Parameters & Arguments
- Return Values
- Scope: Global vs Local
- Closures (Introduction)

Hands-on:

Create a function to add two numbers

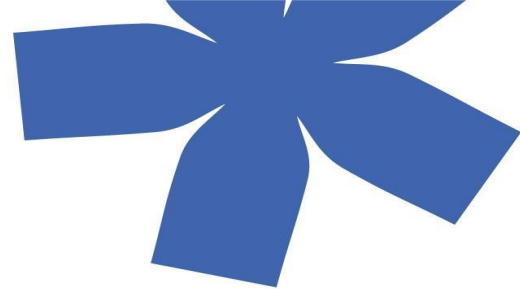
Use arrow functions to simplify code

Exercise:

Write a function to check if a string is a palindrome



OPTIMUM PARTNERS



Day 9: Arrays & Array Methods

Topics:

- Introduction to Arrays
- Array Methods: push, pop, shift, unshift, slice, splice, concat, join
- Iterating Over Arrays: for, forEach, map, filter, reduce

Hands-on:

Create an array and perform various operations

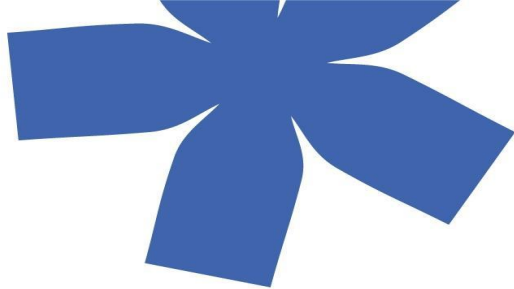
Use map and filter to manipulate arrays

Exercise:

Write a program to find the largest number in an array



OPTIMUM PARTNERS



Day 10: Objects & Prototypes

Topics:

- Introduction to Objects
- Object Properties & Methods
- this Keyword
- Object Constructors & Prototypes
- ES6 Classes

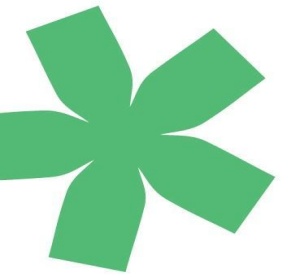
Hands-on:

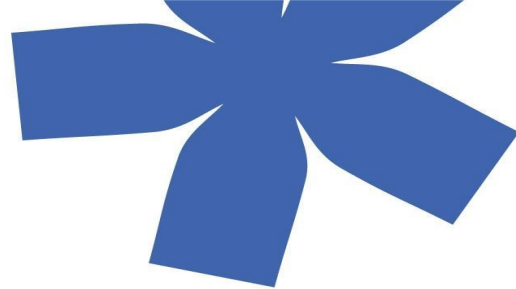
Create an object representing a car with properties and methods

Use classes to create objects

Exercise:

Create a class Person with properties name and age, and a method to display details





Day 11: DOM Manipulation & Events

Topics:

- Introduction to the Document Object Model (DOM)
- Selecting Elements: getElementById, querySelector, querySelectorAll
- Manipulating Elements: innerHTML, textContent, style, classList
- Event Handling: addEventListener, Common Events (click, mouseover, keydown)

Hands-on:

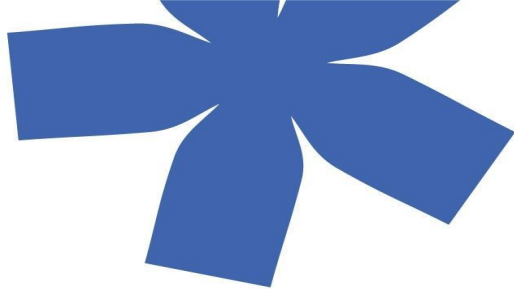
Create a simple HTML page and manipulate it using JavaScript
Add event listeners to buttons

Exercise:

Build a simple to-do list where users can add and remove items



OPTIMUM PARTNERS



Day 12: Error Handling & Debugging

Topics:

- Types of Errors: Syntax, Runtime, Logical
- try, catch, finally
- Debugging Tools: Browser DevTools, console.log, debugger

Hands-on:

Write code that intentionally throws an error and handle it

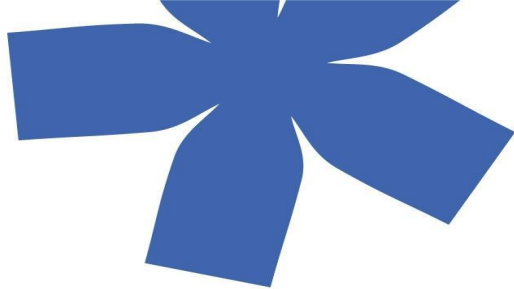
Use browser DevTools to debug code

Exercise:

Write a function that divides two numbers and handles division by zero



OPTIMUM PARTNERS



Day 13: ES6+ Features

Topics:

- Template Literals
- Destructuring Assignment
- Default Parameters
- Rest & Spread Operators
- Modules: import and export

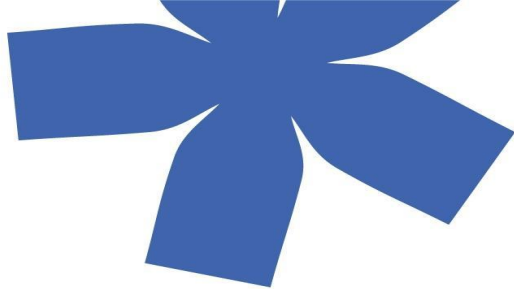
Hands-on:

Use template literals to create dynamic strings
Destructure arrays and objects

Exercise:

Write a function that uses default parameters and the rest operator





Day 14: Asynchronous JavaScript

Topics:

- Introduction to Asynchronous Programming
- Callbacks
- Promises: then, catch, finally
- async & await
- Fetch API

Hands-on:

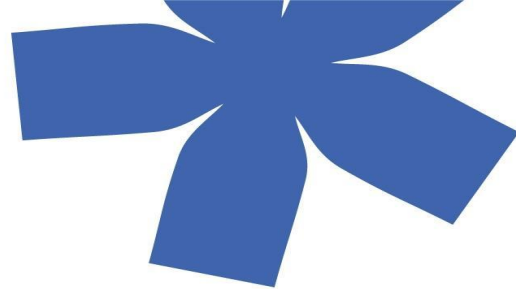
Use fetch to get data from an API

Convert callback-based code to use promises and async/await

Exercise:

Fetch data from a public API and display it on a webpage





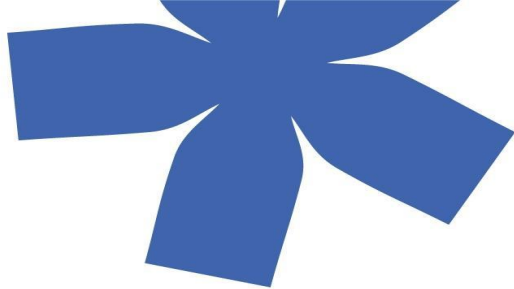
Day 15: Regular Expressions, Cookies & Local Storage-part1

Topics:

- Introduction to Regular Expressions
- Pattern Matching
- Common Use Cases: Validation, Search & Replace
- Cookies:
- What are Cookies?
- Creating, Reading, and Deleting Cookies
- Cookie Attributes: expires, path, domain, secure
- Web Storage: localStorage vs sessionStorage
- Storing & Retrieving Data



OPTIMUM PARTNERS



Day 15: Regular Expressions, Cookies & Local Storage-part2

Hands-on:

Use regular expressions to validate an email address

Create, read, and delete cookies using JavaScript

Store user preferences in localStorage

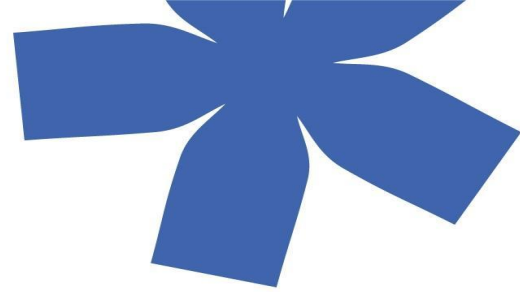
Exercise:

Write a function to validate a phone number using regex.

Build a simple app that saves user input to localStorage and uses cookies to remember user preferences.



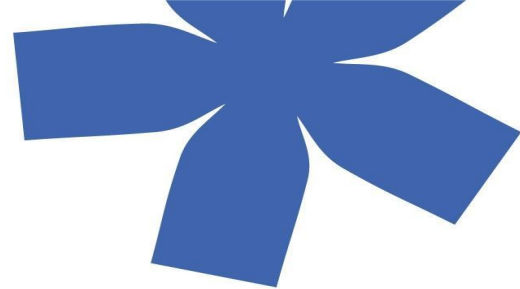
OPTIMUM PARTNERS



Final Projects (Around 2 Days)



OPTIMUM PARTNERS



1. Personal Portfolio Website

A digital showcase of your skills, projects, and professional experience. Features an about section, project gallery, and contact form. Responsive design for all devices.

Core Features:

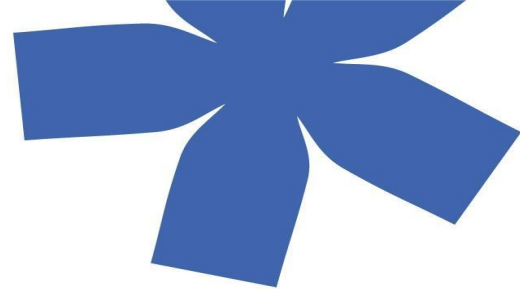
- Homepage with a brief introduction.
- Projects section with descriptions and links.
- Contact form with basic validation.
- Responsive design using Flexbox/Grid.

Advanced Features (Optional):

- Dark/light mode toggle using CSS variables and JavaScript.
- Animations for hover effects or page transitions.
- Integration with a backend to save contact form submissions.



OPTIMUM PARTNERS



2. To-Do List App

A simple app to organize tasks. Users can add, edit, delete, and mark tasks as complete. Responsive and intuitive for daily use.

Core Features:

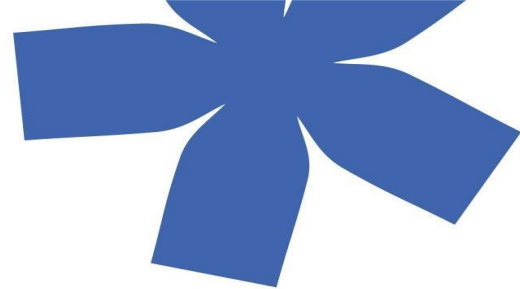
- Add, edit, and delete tasks.
- Mark tasks as complete.
- Responsive design with a clean UI.

Advanced Features (Optional):

- Filter tasks (e.g., show all, completed, or pending).
- Save tasks to localStorage.
- Add due dates and priority levels.



OPTIMUM PARTNERS



3. Event Countdown Timer

A timer that counts down to a specific event. Displays days, hours, minutes, and seconds. Customizable and responsive.

Core Features:

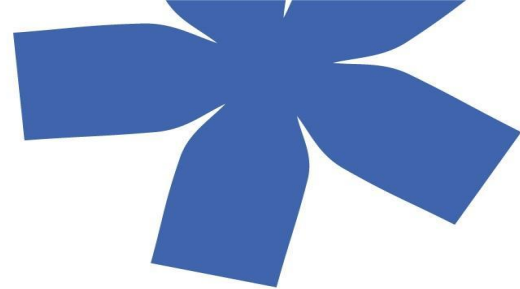
- Display a countdown to a specific event (e.g., New Year).
- Show days, hours, minutes, and seconds remaining.
- Responsive design.

Advanced Features (Optional):

- Allow users to set a custom event date.
- Save the event date in localStorage.
- Add animations for the countdown.



OPTIMUM PARTNERS



4. Expense Tracker

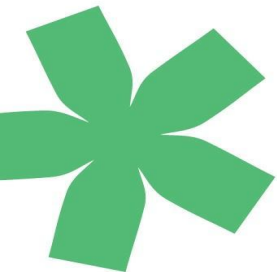
A tool to log and monitor expenses. Users can add expenses, view totals, and categorize spending. Simple and responsive.

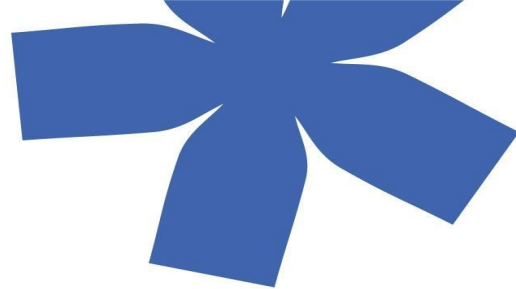
Core Features:

- Add and delete expense entries.
- Display a list of expenses and the total amount.
- Responsive design.

Advanced Features (Optional):

- Categorize expenses (e.g., food, travel, entertainment).
- Save expense data to localStorage.
- Generate a chart to visualize spending.





5. Interactive Photo Gallery

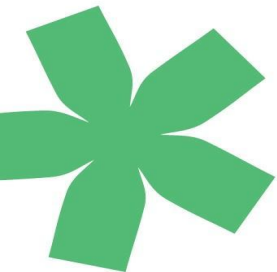
A visually engaging gallery to display images. Features a lightbox for full-size viewing and optional filters. Responsive and user-friendly.

Core Features:

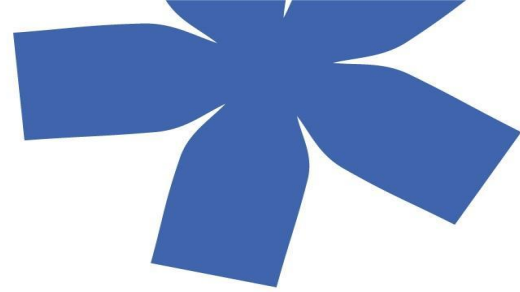
- Display a grid of images.
- Lightbox effect to view images in full size.
- Responsive design.

Advanced Features (Optional):

- Add filters to sort images by category.
- Save favorite images using localStorage.
- Add animations for image transitions.

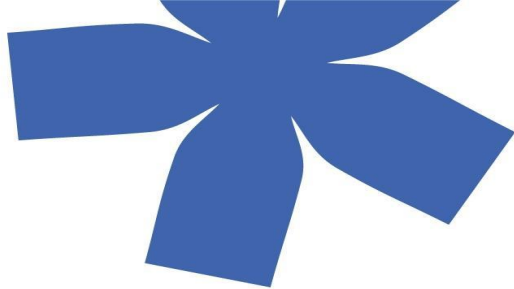


OPTIMUM PARTNERS



TypeScript





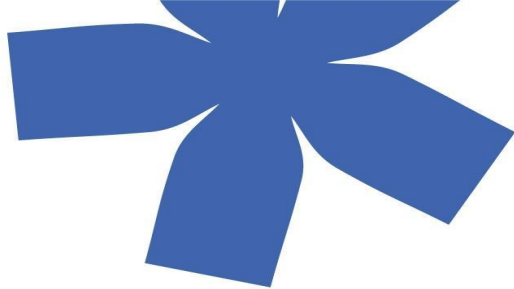
Day 18: TypeScript Basics and Core Concepts - part1

Topics:

- **Introduction to TypeScript**
 - What is TypeScript?
 - Why use TypeScript? (Benefits over JavaScript)
 - Setting up TypeScript (Installing Node.js, TypeScript compiler, and VS Code)
 - Compiling TypeScript to JavaScript (tsc command)
- **Basic Types**
 - Primitive Types: string, number, boolean, null, undefined, void
 - Arrays and Tuples
 - Enums (enum)
 - Any, Unknown, and Never types



OPTIMUM PARTNERS

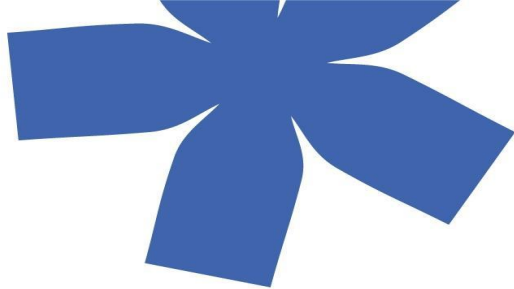


Day 18: TypeScript Basics and Core Concepts - part2

Topics:

- **Type Annotations and Type Inference**
 - Explicit vs Implicit Typing
 - Type Inference in TypeScript
- **Functions in TypeScript**
 - Function Parameter and Return Type Annotations
 - Optional and Default Parameters
 - Rest Parameters
 - Arrow Functions





Day 18: TypeScript Basics and Core Concepts - part3

Hands-on:

Set up a TypeScript project and compile a .ts file to .js

Create variables with different types and use type annotations

Write functions with optional, default, and rest parameters

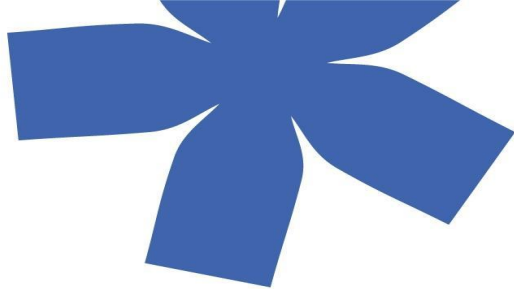
Exercise:

Create a TypeScript program that calculates the area of a rectangle using type annotations

Write a function that takes a variable number of arguments (using rest parameters) and returns their sum

Create an enum for days of the week and print the current day





Day 18: TypeScript Basics and Core Concepts - part4

Hands-on:

Set up a TypeScript project and compile a .ts file to .js

Create variables with different types and use type annotations

Write functions with optional, default, and rest parameters

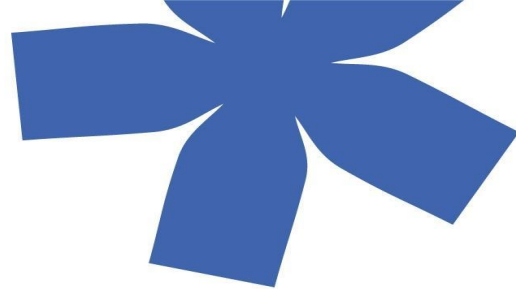
Exercise:

Create a TypeScript program that calculates the area of a rectangle using type annotations

Write a function that takes a variable number of arguments (using rest parameters) and returns their sum

Create an enum for days of the week and print the current day

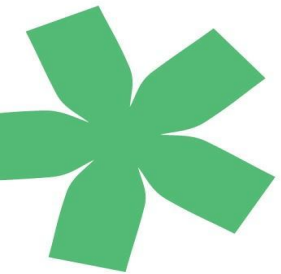


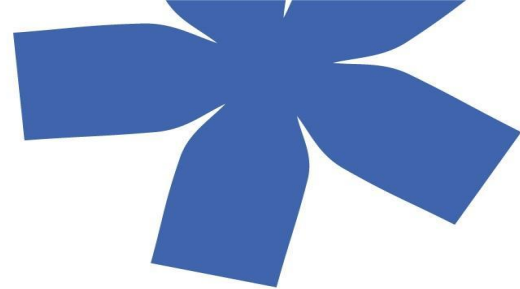


Day 19: Advanced Types and Object-Oriented Programming - part1

Topics:

- **Advanced Types**
 - Union and Intersection Types
 - Type Aliases and Custom Types
 - Literal Types
 - Type Assertions (as keyword)
- **Interfaces**
 - Defining Interfaces
 - Optional and Readonly Properties
 - Extending Interfaces
 - Interface vs Type Aliases



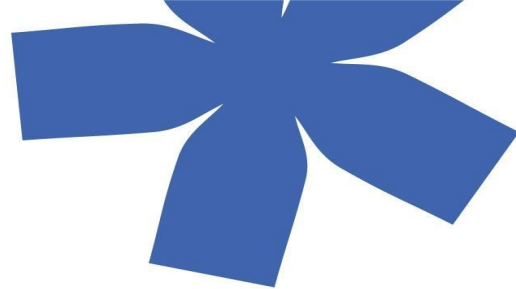


Day 19: Advanced Types and Object-Oriented Programming - part2

Topics:

- **Classes and Object-Oriented Programming**
 - Class Syntax
 - Constructors and Properties
 - Access Modifiers (public, private, protected)
 - Getters and Setters
 - Inheritance and Method Overriding
 - Abstract Classes





Day 19: Advanced Types and Object-Oriented Programming - part3

Hands-on:

- Create interfaces for objects like User or Product and implement them

- Use union and intersection types to handle complex data structures

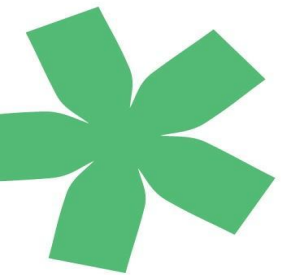
- Build a class hierarchy (e.g., Animal -> Dog and Cat) with inheritance

Exercise:

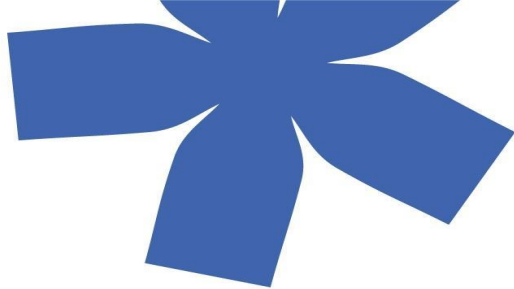
- Create an interface Person with properties like name, age, and email. Implement this interface in a class Employee

- Write a program that uses union types to handle different types of inputs (e.g., string or number)

- Create an abstract class Shape with an abstract method `calculateArea()`. Extend it with classes like Circle and Rectangle



OPTIMUM PARTNERS

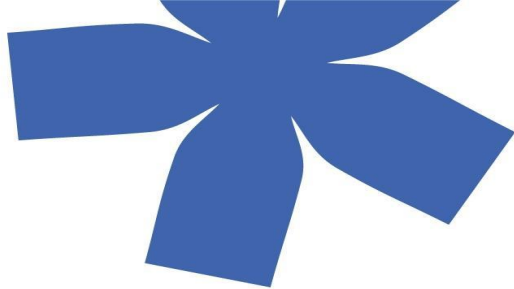


Day 20: Generics, Utility Types, and Modules - part1

Topics:

- **Generics**
 - Introduction to Generics
 - Generic Functions and Classes
 - Generic Constraints
 - Using Generics with Interfaces
- **Utility Types**
 - Common Utility Types: Partial, Required, Readonly, Record, Pick, Omit
 - Mapped Types
- **Modules**
 - Importing and Exporting Modules
 - Default vs Named Exports
 - Organizing Code with Modules
- **Namespaces**
 - Introduction to Namespaces
 - Namespaces vs Modules





Day 20: Generics, Utility Types, and Modules - part2

Hands-on:

- Write a generic function to handle arrays of any type

- Use utility types like `Partial` and `Pick` to manipulate object types

- Organize a TypeScript project using modules and namespaces

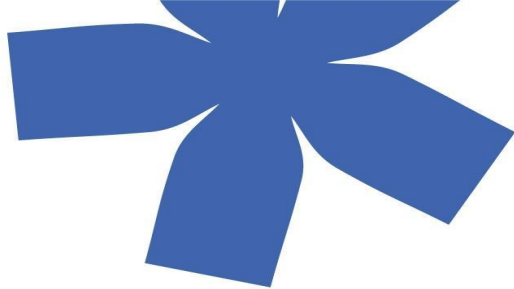
Exercise:

- Create a generic function `reverseArray` that reverses an array of any type

- Use the `Pick` utility type to create a new type with only specific properties from an existing interface

- Build a small project with multiple modules (e.g., `math.ts`, `utils.ts`) and import them into a main file



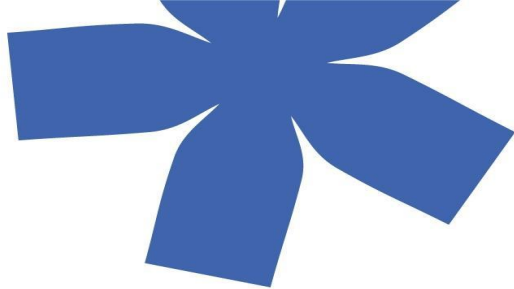


Day 21: Advanced TypeScript Features and Tooling - part1

Topics:

- **Decorators**
 - Introduction to Decorators
 - Class, Method, Property, and Parameter Decorators
 - Built-in Decorators (@sealed, @override)
- **TypeScript Configuration**
 - tsconfig.json File
 - Key Compiler Options (target, module, strict, outDir, etc.)
- **Working with Third-Party Libraries**
 - Using DefinitelyTyped and @types Packages
 - TypeScript with React, Angular, or Node.js (Overview)
- **Debugging and Testing**
 - Debugging TypeScript in VS Code
 - Writing Unit Tests with TypeScript (Jest or Mocha)





Day 21: Advanced TypeScript Features and Tooling - part1

Hands-on:

- Create and use a class decorator to log method calls

- Configure a TypeScript project using tsconfig.json

- Install and use a third-party library with TypeScript types (e.g., lodash)

Exercise:

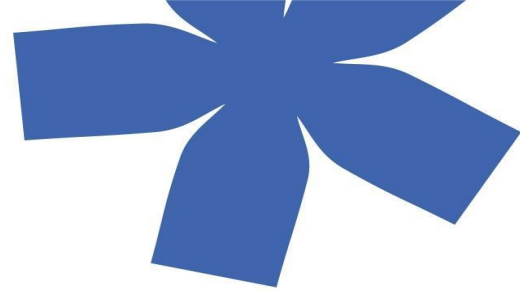
- Write a class decorator that logs the execution time of a method

- Configure a TypeScript project to output ES6 modules and enable strict type-checking

- Write a simple unit test for a TypeScript function using Jest

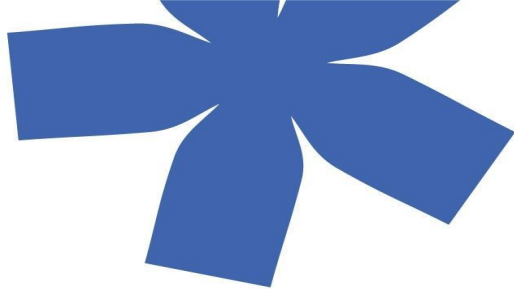


OPTIMUM PARTNERS



Angular





Day 22: Introduction to Angular

Topics:

- What is Angular? (Overview, History, and Evolution)
- Angular vs AngularJS
- Setting Up the Angular Environment (Node.js, npm, Angular CLI)
- Creating Your First Angular Application (ng new)
- Angular Project Structure (src, app, components, modules, etc.)
- Angular CLI Commands (ng serve, ng generate, ng build)
- Angular 18 Feature: Angular CLI Enhancements (e.g., faster builds, improved error messages)

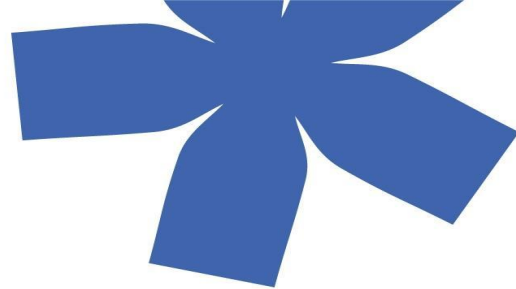
Hands-on:

Install Angular CLI and create a new Angular project
Run the application and explore the project structure
Use Angular CLI to generate a new component

Exercise:

Create a new Angular project and add a "Welcome" component that displays a greeting message





Day 23: Angular Components and Templates

Topics:

- What are Components? (Component Architecture)
- Creating Components (ng generate component)
- Component Metadata (@Component Decorator: selector, template, styles)
- Templates and Interpolation ({{ }})
- Property Binding ([]) and Event Binding (())
- Two-Way Data Binding ([[(ngModel)])
- Angular 18 Feature: Standalone Components

Hands-on:

Create a component with a template and use interpolation to display data

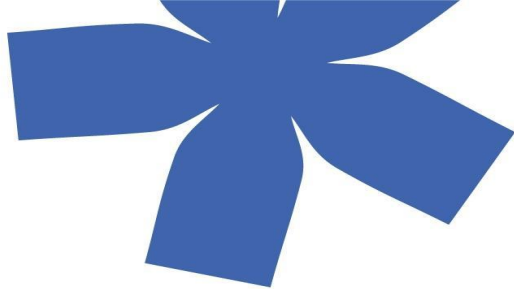
Bind a button click event to a method in the component class

Use ngModel to implement two-way data binding in a form

Exercise:

Build a simple user profile component with input fields for name and email, and display the entered data using interpolation





Day 24: Directives and Pipes

Topics:

- Built-in Directives (ngIf, ngFor, ngSwitch)
- Attribute Directives (ngClass, ngStyle)
- Custom Directives (Creating and Using)
- Built-in Pipes (date, uppercase, lowercase, currency, etc.)
- Custom Pipes (Creating and Using)

Hands-on:

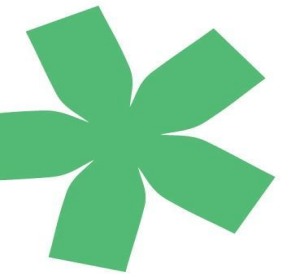
Use ngFor to display a list of items

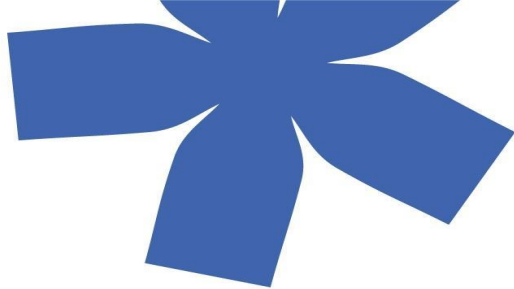
Create a custom directive that changes the background color of an element

Use pipes to format dates and currency values

Exercise:

Build a component that displays a list of products with their prices formatted using a custom currency pipe





Day 25: Angular Modules and Dependency Injection

Topics:

- What are Angular Modules? (@NgModule)
- Declarations, Imports, Exports, and Providers
- Lazy Loading Modules
- Dependency Injection (DI) in Angular
- Services and Injectables (@Injectable)
- Hierarchical Injectors (Root, Module, Component)
- Angular 19 Feature: Enhanced Dependency Injection

Hands-on:

Create a feature module and lazy load it

Create a service and inject it into a component

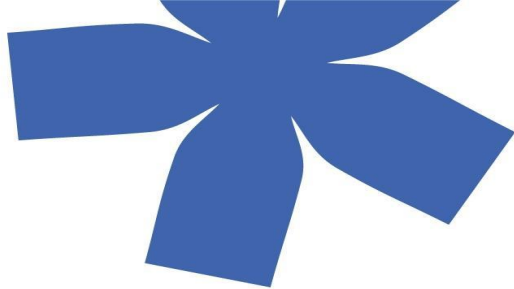
Use providedIn: 'root' to make a service available globally

Exercise:

Build a multi-module Angular application with a lazy-loaded feature module

Create a service to fetch and display data from a mock API





Day 26: Angular Routing and Navigation

Topics:

- Setting Up Routes (RouterModule.forRoot)
- Router Outlet and Router Links
- Route Parameters (Dynamic Routes)
- Child Routes and Nested Routes
- Route Guards (CanActivate, CanDeactivate, CanLoad)
- Lazy Loading with Routes
- Angular 18 Feature: Improved Debugging Tools for Routing

Hands-on:

Set up routing in an Angular application.

Create a route with parameters and access them in a component.

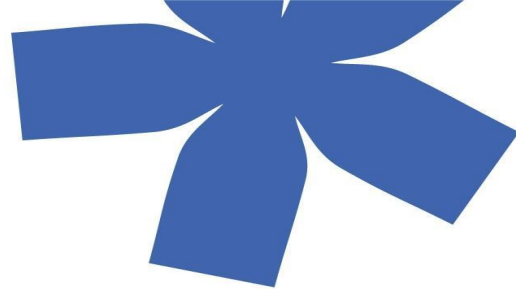
Implement a route guard to restrict access to a route.

Exercise:

Build a multi-page application with routes for a home page, about page, and contact page

Implement a route guard to protect a "dashboard" route





Day 27: Angular Forms (Template-Driven and Reactive)

Topics:

- Template-Driven Forms (ngForm, ngModel)
- Reactive Forms (FormGroup, FormControl, FormBuilder)
- Form Validation (Built-in and Custom Validators)
- Dynamic Forms (Adding/Removing Form Controls)
- FormArray and Nested Forms
- Angular 18 Feature: Enhanced Forms API

Hands-on:

Create a template-driven form with validation

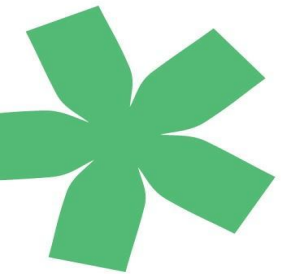
Build a reactive form with dynamic form controls

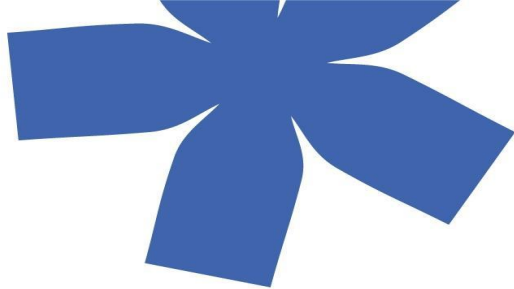
Implement custom validation for a form field

Exercise:

Build a registration form using reactive forms with validation for email, password, and confirm password

Create a dynamic form where users can add multiple addresses





Day 28: Angular HTTP Client and Services

Topics:

- Introduction to Angular HTTP Client (HttpClientModule)
- Making GET, POST, PUT, DELETE Requests
- Handling HTTP Errors (catchError, retry)
- Interceptors (Request and Response Interceptors)
- Using RxJS Operators (map, switchMap, mergeMap, etc.)

Hands-on:

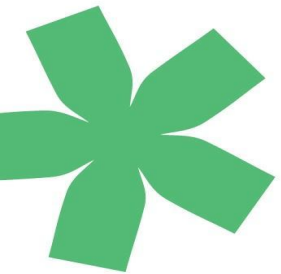
Fetch data from a public API and display it in a component

Create an HTTP interceptor to add headers to every request

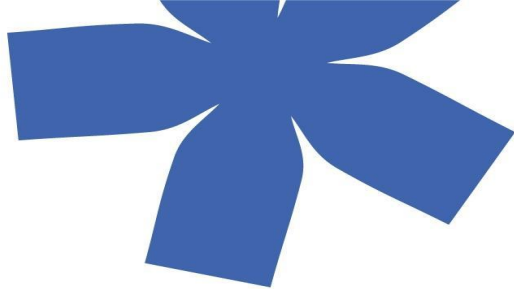
Handle errors in HTTP requests using RxJS operators

Exercise:

Build a simple CRUD application using Angular HTTP Client to interact with a mock API



OPTIMUM PARTNERS



Day 29: State Management with NgRx

Topics:

- Introduction to State Management
- NgRx Store (Actions, Reducers, Selectors)
- Effects (Side Effects in NgRx)
- Entity Management with NgRx
- Debugging NgRx with Redux DevTools

Hands-on:

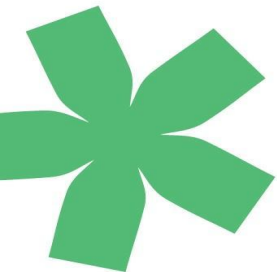
Set up NgRx in an Angular application

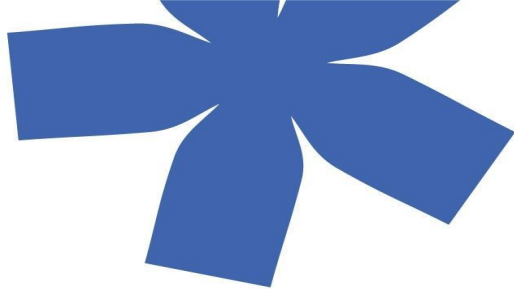
Create actions, reducers, and selectors for a simple feature

Use effects to handle asynchronous operations

Exercise:

Build a simple to-do list application using NgRx for state management





Day 30: Angular Animations

Topics:

- Introduction to Angular Animations (`BrowserAnimationsModule`)
- Trigger, State, and Transition
- Keyframes and Animation Timing
- Route Animations (Page Transitions)

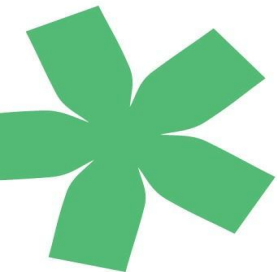
Hands-on:

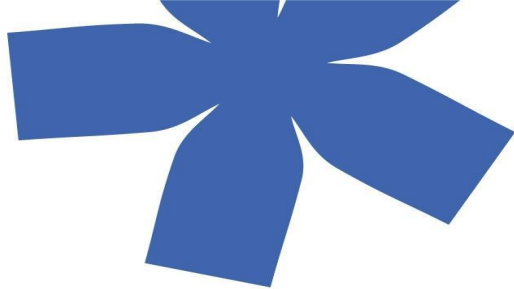
Create a simple animation for a button hover effect

Implement route animations for page transitions

Exercise:

Build a component with a toggle animation that expands and collapses content





Day 31: Angular Universal (Server-Side Rendering)

Topics:

- What is Angular Universal?
- Setting Up Angular Universal
- Server-Side Rendering (SSR) vs Client-Side Rendering (CSR)
- SEO Benefits of SSR
- Deploying an Angular Universal Application

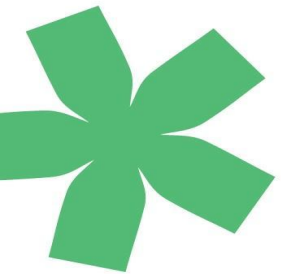
Hands-on:

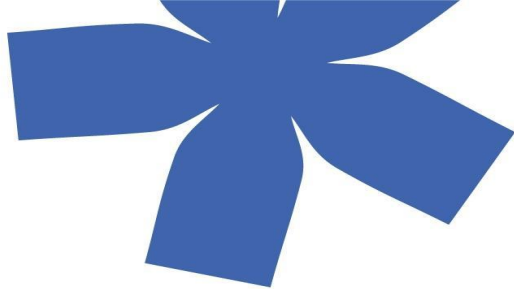
Set up Angular Universal in an existing Angular application

Render a simple Angular application on the server

Exercise:

Convert an existing Angular application to use Angular Universal





Day 32: Testing in Angular

Topics:

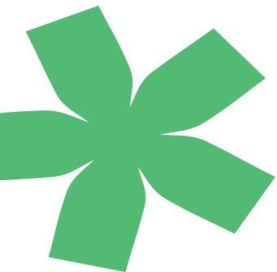
- Introduction to Testing in Angular (Jasmine, Karma)
- Unit Testing Components, Services, and Directives
- Testing Asynchronous Code
- Mocking Dependencies in Tests

Hands-on:

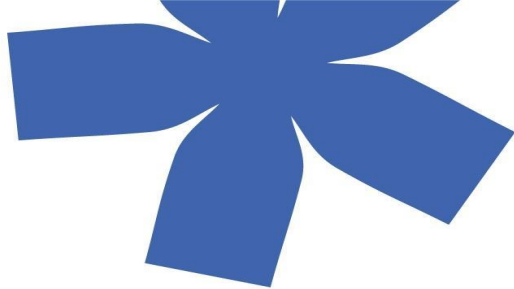
Write unit tests for a simple component and service

Exercise:

Write unit tests for a form component and an HTTP service



OPTIMUM PARTNERS



Day 33: Deployment and Optimization

Topics:

- Building an Angular Application for Production (`ng build --prod`)
- Optimizing Angular Applications (Lazy Loading, Ahead-of-Time Compilation)

Hands-on:

Build Angular application

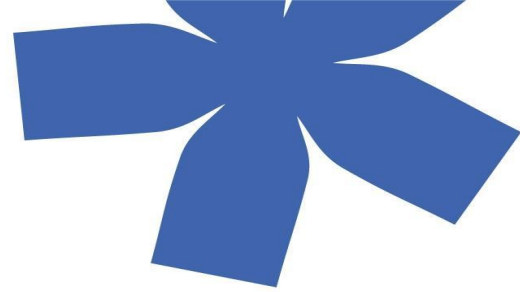
Use Angular DevTools to analyze performance

Exercise:

Optimize a simple Angular application for performance and generate a production build



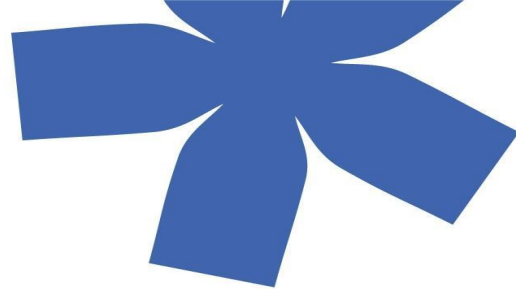
OPTIMUM PARTNERS



Final Projects (Around 4 Days)



OPTIMUM PARTNERS



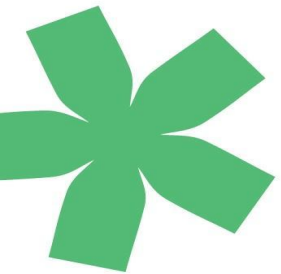
1. E-Commerce Web Application

Core Features:

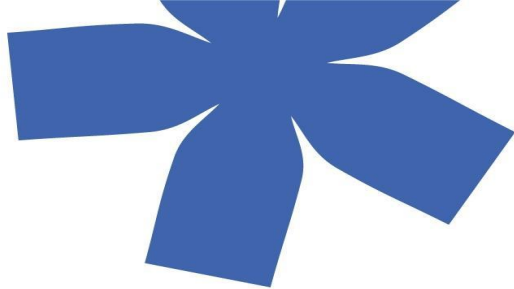
- **Components and Templates:** Create product listing, product details, and shopping cart components.
- **Routing:** Set up routes for home, product details, and checkout pages.
- **Forms:** Implement a checkout form using reactive forms with validation.
- **HTTP Client:** Fetch product data from a mock API (e.g., JSON Server or a public API like FakeStoreAPI).
- **State Management:** Use NgRx to manage the shopping cart state (add/remove items, calculate total).
- **Pipes:** Format product prices using a custom currency pipe.

Advanced Features (Optional):

- **Angular Universal:** Implement server-side rendering for better SEO.
- **Animations:** Add animations for adding items to the cart or transitioning between pages.
- **Route Guards:** Protect the checkout route so users must be logged in.
- **Interceptors:** Add an HTTP interceptor to include an authentication token in API requests.



OPTIMUM PARTNERS



2. Online Learning Platform

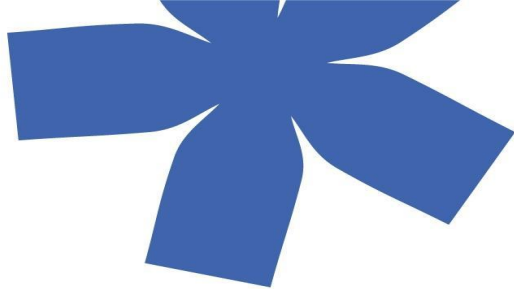
Core Features:

- **Components:** Create components for course listings, course details, and enrollment.
- **Routing:** Set up routes for home, course details, and enrollment form.
- **Forms:** Use reactive forms for course enrollment.
- **HTTP Client:** Fetch course data from a mock API.
- **State Management:** Use NgRx to manage enrolled courses.
- **Pipes:** Create a custom pipe to filter courses by category or difficulty.

Advanced Features (Optional):

- **Route Guards:** Protect the enrollment route so users must log in.
- **Animations:** Add animations for course enrollment confirmation.
- **Interceptors:** Add an interceptor to handle API errors during enrollment.
- **Angular Universal:** Implement SSR for better SEO on course pages.





3. Job Board Application

Core Features:

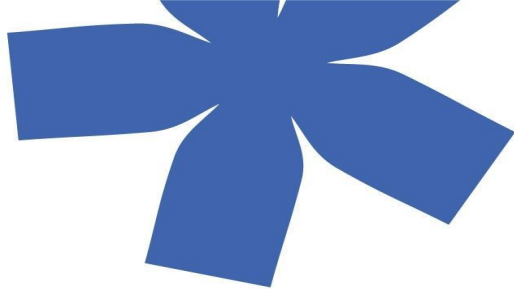
- **Components:** Create components for job listings, job details, and job application.
- **Routing:** Set up routes for home, job details, and application form.
- **Forms:** Use reactive forms for applying to jobs.
- **HTTP Client:** Fetch job data from a mock API.
- **State Management:** Use NgRx to manage job applications.
- **Pipes:** Create a custom pipe to filter jobs by category or location.

Advanced Features (Optional):

- **Route Guards:** Protect the application route so users must log in.
- **Animations:** Add animations for applying to jobs.
- **Angular Universal:** Implement SSR for better SEO on job listings.
- **Testing:** Write unit tests for the job service and components.



OPTIMUM PARTNERS



4. Fitness Tracker

Core Features:

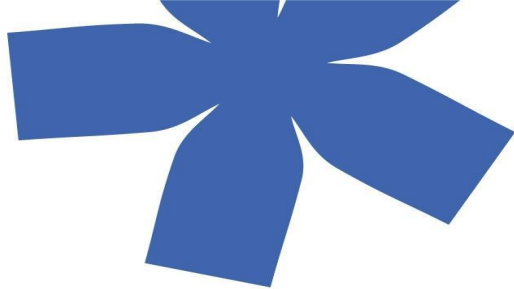
- **Components:** Build components for workout list, workout details, and workout creation.
- **Routing:** Set up routes for home, workout details, and a dashboard.
- **Forms:** Use reactive forms for adding and editing workouts.
- **HTTP Client:** Save workouts to a mock backend (e.g., JSON Server).
- **State Management:** Use NgRx to manage workouts and progress.
- **Pipes:** Create a custom pipe to format workout durations.

Advanced Features (Optional):

- **Route Guards:** Protect the dashboard route so users must log in.
- **Animations:** Add animations for completing workouts.
- **Interceptors:** Add an interceptor to handle authentication tokens for API requests.
- **Testing:** Write unit tests for the workout service and components.



OPTIMUM PARTNERS



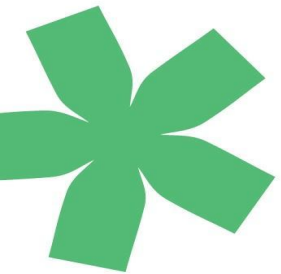
5. Travel Booking App

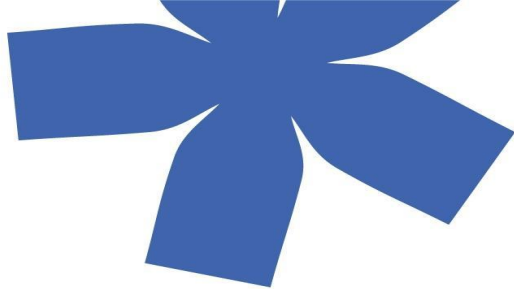
Core Features:

- **Components:** Create components for flight/hotel listings, booking details, and payment.
- **Routing:** Set up routes for home, booking details, and payment form.
- **Forms:** Use reactive forms for booking and payment.
- **HTTP Client:** Fetch flight/hotel data from a mock API.
- **State Management:** Use NgRx to manage bookings.
- **Pipes:** Create a custom pipe to filter flights/hotels by price or rating.

Advanced Features (Optional):

- **Route Guards:** Protect the payment route so users must log in.
- **Animations:** Add animations for booking confirmation.
- **Interceptors:** Add an interceptor to handle API errors during booking.
- **Angular Universal:** Implement SSR for better SEO on travel listings.





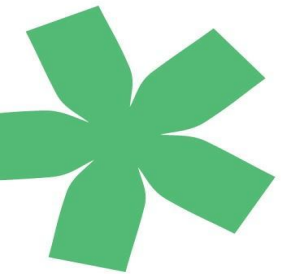
6. Real Estate Listing App

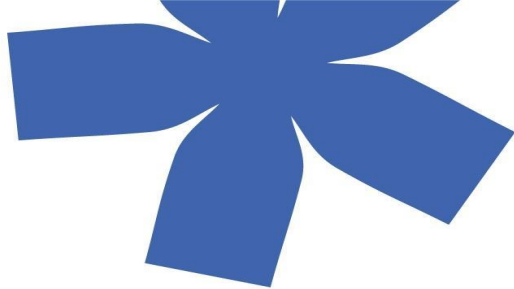
Core Features:

- **Components:** Build components for property listings, property details, and contact forms.
- **Routing:** Set up routes for home, property details, and contact form.
- **Forms:** Use reactive forms for contacting property agents.
- **HTTP Client:** Fetch property data from a mock API.
- **State Management:** Use NgRx to manage favorite properties.
- **Pipes:** Create a custom pipe to filter properties by price or location.

Advanced Features (Optional):

- **Route Guards:** Protect the contact form route so users must log in.
- **Animations:** Add animations for adding properties to favorites.
- **Interceptors:** Add an interceptor to handle API errors during property searches.
- **Testing:** Write unit tests for the property service and components.





7. Online Voting System

Core Features:

- **Components:** Build components for poll listings, poll details, and voting.
- **Routing:** Set up routes for home, poll details, and results.
- **Forms:** Use reactive forms for submitting votes.
- **HTTP Client:** Fetch poll data from a mock API.
- **State Management:** Use NgRx to manage poll results.
- **Pipes:** Create a custom pipe to format poll results.

Advanced Features (Optional):

- **Route Guards:** Protect the voting route so users must log in.
- **Animations:** Add animations for vote submission.
- **Interceptors:** Add an interceptor to handle API errors during voting.
- **Testing:** Write unit tests for the poll service and components.

