Day 1: HTML Fundamentals

* Learn the basics of HTML, including tags, attributes, and document structure.
* Task: Build a simple webpage with proper semantic HTML markup.

Day 2: CSS Fundamentals

* Dive into CSS properties, selectors, and box model concepts.
* Task: Style your previously created webpage using CSS.

Day 3: CSS Layouts

* Explore different CSS layout techniques, such as float, flexbox, and grid.
* Task: Create a responsive layout for your webpage using flexbox.

Day 4: JavaScript Basics

* Learn JavaScript fundamentals, including variables, data types, functions, and control flow.
* Task: Write a JavaScript function to calculate and display the current date.

Day 5: DOM Manipulation

* Understand how to interact with the Document Object Model (DOM) using JavaScript.
* Task: Create an interactive form that validates user input using JavaScript.

Day 6: JavaScript Events

* Learn about event handling and how to respond to user actions.
* Task: Add event listeners to your form elements to enhance interactivity.

Day 7: Responsive Web Design

* Explore responsive design principles and techniques.
* Task: Make your webpage responsive by using media queries.

Day 8: CSS Preprocessors (e.g., Sass or Less)

* Learn how to use a CSS preprocessor to enhance your styling workflow.
* Task: Convert your CSS code to a preprocessor of your choice.

Day 9: Version Control (e.g., Git)

* Understand the basics of version control to manage your code effectively.
* Task: Create a Git repository and make commits for your webpage project.

Day 10: Front-End Frameworks (e.g., React or Angular)

* Explore a popular front-end framework of your choice.
* Task: Create a simple project using the chosen framework and showcase your understanding.

Day 11: Advanced JavaScript Concepts

* Dive deeper into JavaScript with topics like closures, prototypes, and asynchronous programming.
* Task: Build a small JavaScript application that fetches data from an API.

Day 12: Web Performance Optimization

* Learn techniques to optimize website performance, such as minification and lazy loading.
* Task: Optimize your webpage's loading time and measure the improvements.

Day 13: CSS Animation and Transitions

* Explore CSS animations and transitions to create engaging user experiences.
* Task: Add animations to your webpage elements to enhance visual appeal.

Day 14: Accessibility Best Practices

* Understand how to make your websites accessible to all users.
* Task: Implement accessibility features and test your webpage using accessibility tools.

Day 15: Project Showcase and Portfolio Development

* Review and refine your previous projects, focusing on code quality and presentation.
* Task: Create a professional portfolio website to showcase your projects and skills.

Days 16-60: Advanced Topics and Interview Preparation

* Dive deeper into advanced front-end concepts based on your chosen framework.
* Learn about state management, component architecture, performance optimization, and more.
* Task: Complete a series of challenging projects that simulate real-world scenarios.
* Practice technical interview questions, algorithms, and problem-solving for front-end development roles

| **Day** | **Task** | **Topics to Cover** | **Resources** |
| --- | --- | --- | --- |
| 1 | Set up Angular development environment | Installing Node.js and Angular CLI | Official Angular documentation, online tutorials |
| 2 | Create your first Angular app | Angular project structure, components | Angular documentation, tutorials, YouTube videos |
| 3 | Understanding Angular components | Template, class, metadata, data binding | Angular documentation, tutorials, YouTube videos |
| 4 | Working with Angular templates | Directives, interpolation, property binding | Angular documentation, tutorials, YouTube videos |
| 5 | Styling Angular components | CSS classes, component styles, style encapsulation | Angular documentation, tutorials, YouTube videos |
| 6 | Exploring Angular modules | NgModule, feature modules, lazy loading | Angular documentation, tutorials, YouTube videos |
| 7 | Understanding Angular services | Dependency injection, singleton services | Angular documentation, tutorials, YouTube videos |
| 8 | Working with Angular forms | Template-driven forms, form validation | Angular documentation, tutorials, YouTube videos |
| 9 | Implementing reactive forms in Angular | FormControl, FormGroup, form validation | Angular documentation, tutorials, YouTube videos |
| 10 | Routing in Angular | Set up routes, route parameters, navigation | Angular documentation, tutorials, YouTube videos |
| 11 | Fetching data with Angular's HttpClient | Making HTTP requests, handling responses | Angular documentation, tutorials, YouTube videos |
| 12 | Angular component communication | @Input, @Output, EventEmitter | Angular documentation, tutorials, YouTube videos |
| 13 | Introduction to Angular pipes | Built-in pipes, creating custom pipes | Angular documentation, tutorials, YouTube videos |
| 14 | Handling errors and exceptions | Error handling, error interceptors | Angular documentation, tutorials, YouTube videos |
| 15 | Unit testing Angular components | Writing and running tests using Jasmine | Angular documentation, testing guides, tutorials |
| 16 | Deploying an Angular app | Building and deploying your app | Angular documentation, tutorials, deployment guides |
| 17-60 | Building a front-end Angular website | Continuously develop your website project | Apply learned concepts to build your own Angular app |

| **Day** | **Task** | **Topics to Cover** | **Resources** |
| --- | --- | --- | --- |
| 1-5 | HTML Basics | HTML tags, elements, attributes, semantic structure, forms | Online tutorials, w3schools.com, MDN Web Docs |
| 6-10 | CSS Fundamentals | Selectors, properties, positioning, layout, responsive design | Online tutorials, w3schools.com, MDN Web Docs |
| 11-15 | Bootstrap Framework Basics | Grid system, responsive utilities, components | Official Bootstrap documentation, tutorials, examples |
| 16-20 | Intermediate HTML and CSS Concepts | CSS Flexbox, CSS Grid, CSS transitions and animations | Online tutorials, CSS-Tricks, Codecademy |
| 21-30 | Angular Basics and Project Setup | Review Angular fundamentals, create a new Angular project, set up routing | Angular documentation, tutorials, YouTube videos |
| 31-35 | Angular Material Design and UI Components | Install and configure Angular Material, use Material components and styles | Angular Material documentation, tutorials, examples |
| 36-40 | Advanced Angular Components and Directives | Custom component communication, ViewChild, ContentChild, structural directives, attribute directives | Angular documentation, tutorials, YouTube videos |
| 41-45 | Advanced Angular Forms | Reactive forms with complex validation, dynamic forms, form arrays | Angular documentation, tutorials, YouTube videos |
| 46-50 | Advanced Angular Routing and Guards | Child routes, lazy loading modules, route guards | Angular documentation, tutorials, YouTube videos |
| 51-55 | Working with HTTP Interceptors | Intercepting and modifying HTTP requests and responses | Angular documentation, tutorials, YouTube videos |
| 56-60 | Angular Testing Techniques | Unit testing with TestBed and Jasmine, component testing, test doubles | Angular documentation, testing guides, tutorials |
| 61-75 | Project Development | Plan, design, and build your front-end Angular website project | Apply learned concepts, leverage online resources |
| 76-90 | Project Refinement and Deployment | Improve user experience, optimize performance, deploy your Angular website project to a hosting platform | Apply best practices, documentation, deployment guides |

During the project development phase (days 61-75), focus on the following key topics:

1. Responsive design: Ensure your website looks great and functions well across different devices and screen sizes.
2. Angular components: Create reusable and modular components for different sections of your website.
3. Data management: Implement services and utilize Angular's HttpClient to fetch data from APIs and display it on your website.
4. User interaction: Incorporate user input forms and handle form submissions and validations.
5. Routing and navigation: Set up multiple routes within your application, enabling users to navigate between different pages or sections.
6. UI/UX considerations: Pay attention to user experience, accessibility, and visual aesthetics to create an engaging website.