DEVSLOOP

Intern Training Document

September 16, 2024

Training document for Full Stack Dev

Learning Ruby on Rails is like discovering the magic behind web development.

Note:

We'll have daily discussions & we'll assume that you will be learning the concepts by practical implementation yourselves.

Rule of thumb: Consider Google your best friend.

Week 1: HTML & CSS Practice

Day 1: Introduction to HTML

- **Objective:** Learn the basics of HTML structure and semantic elements.
- **Topics:**
  - HTML document structure (`<html>`, `<head>`, `<body>`)
  - Semantic elements: `<header>`, `<footer>`, `<nav>`, `<article>`, etc.
  - Forms and input elements
- **Exercise:**
  - Create a basic webpage with a header, footer, and a form for user input (name, email, and message fields).

Day 2: Introduction to CSS

- **Objective:** Style HTML elements using CSS.
- **Topics:**
  - CSS syntax and selectors
  - Inline, internal, and external styles
  - Classes and IDs
- **Exercise:**
  - Add styles to the webpage created on Day 1, using external CSS. Style the header, footer, form fields, and buttons.

Day 3: Box Model and Layouts

- **Objective:** Understand the CSS box model and layout techniques.
- **Topics:**
  - Margins, padding, borders, and content
  - Display properties: block, inline-block, and inline
  - Flexbox basics
- **Exercise:**
  - Create a multi-column layout using Flexbox. Arrange the form, header, and footer in a responsive layout.

Day 4: Responsive Design

- **Objective:** Make web pages responsive for different screen sizes.
- **Topics:**
  - Media queries
  - Responsive images
  - Using percentages and relative units for layout

- **Exercise:**
    - Update the previous layout to make it responsive using media queries, adjusting for mobile and tablet views.

Day 5: CSS Grid and Advanced Styling

- **Objective:** Explore advanced CSS techniques with CSS Grid and animations.
- **Topics:**
    - CSS Grid layout system
    - Transitions and animations
    - Hover effects
- **Exercise:**
    - Create a photo gallery using CSS Grid. Add hover effects and transitions to the images for smooth interactivity.

Week 2: JavaScript (DOM Manipulation & Form Validation)

Day 1: Introduction to JavaScript and DOM

- **Objective:** Understand basic JavaScript and manipulate the DOM.
- **Topics:**
    - JavaScript variables, data types, and functions
    - DOM tree and basic manipulation methods (`getElementById`, `querySelector`, etc.)
    - Event handling (`addEventListener`)
- **Exercise:**
    - Add interactivity to the form created in Week 1 by changing the form background color when the user clicks a button.

Day 2: DOM Manipulation & Events

- **Objective:** Learn to handle user events and dynamically manipulate HTML elements.
- **Topics:**
    - Events (click, input, change, submit)
    - Modifying element content and attributes (`innerHTML`, `setAttribute`)
- **Exercise:**
    - Add a “Submit” button to the form. When clicked, display a message thanking the user for submitting the form, replacing the form fields.

Day 3: Forms and Input Validation

- **Objective:** Implement form validation using JavaScript.
- **Topics:**
  - Input validation techniques (required fields, pattern matching, length checks)
  - Displaying error messages
- **Exercise:**
  - Implement validation for the form (name must be 3+ characters, email must be valid, and message cannot be empty). Show error messages next to invalid fields.

Day 4: More DOM Manipulation (Traversing the DOM)

- **Objective:** Learn to traverse the DOM and manipulate sibling/parent/child elements.
- **Topics:**
  - DOM traversal methods (`parentNode`, `childNodes`, `nextElementSibling`, etc.)
  - Creating and removing elements dynamically
- **Exercise:**
  - Allow users to add new form fields dynamically. For example, they can add an additional "phone number" field by clicking an “Add Phone” button.

Day5: Final Project - Form Validation & Interactivity

- **Objective:** Combine all learned concepts to create a fully interactive form.
- **Exercise:**
  - Build a complete form with the following:
    - Dynamic form fields
    - Input validation with error messages
    - A success message upon form submission
    - Basic CSS animations on form validation

Week 3: Ruby Fundamentals and OOP Basics

Day 1: Ruby Syntax and Data Structures

- Objective: Get familiar with basic Ruby syntax and data structures.
- Topics: Variables, Data types (Strings, Arrays, Hashes), Conditionals, Loops.

- Exercise:
    - Write a program that takes an array of numbers and returns a new array containing only the even numbers.
    - Write a method that checks if a string is a palindrome.

Day 2: Methods, Blocks, and Enumerable

- Objective: Understand how to define and use methods, blocks, and Ruby's `Enumerable` module.
- Topics: Defining methods, Method arguments, Yielding to blocks, Common Enumerable methods (`map`, `select`, `reduce`).
- Exercise:
    - Create a method that takes a block and applies it to each element of an array.
    - Write a method using `Enumerable#reduce` to find the sum of an array of numbers.

Day 3: OOP Basics - Classes and Objects

- Objective: Learn about creating classes and objects in Ruby.
- Topics: Classes, Objects, Instance variables, Methods, Initializers (`initialize` method).
- Exercise:
    - Create a `Book` class with attributes like `title`, `author`, and `pages`. Write methods to display book details.
    - Create a `Library` class that can store multiple `Book` objects and has methods to add and list books.

Day 4: OOP - Encapsulation, Inheritance, and Polymorphism

- Objective: Understand advanced OOP concepts like encapsulation, inheritance, and polymorphism.
- Topics: Access control (`public`, `private`), Inheritance, Method overriding, Polymorphism.
- Exercise:
    - Add access control to the `Book` class so certain details are only accessible through methods.
    - Create a `DigitalBook` class that inherits from `Book`, and override the method that displays book details to include file format.

Day 5: Modules, Mixins, and Error Handling

- Objective: Learn how to use modules for mixins and handle exceptions in Ruby.
- Topics: Modules, Mixing in functionality (`include`, `extend`), Error handling (`begin`, `rescue`).
- Exercise:
    - Create a `Borrowable` module that can be mixed into `Book` and `DigitalBook` classes to add functionality for borrowing and returning books.
    - Add error handling to prevent borrowing a book that is already borrowed.

Week 4: Advanced OOP and Gem Creation

Day 1:: Working with Gems and Bundler

- Objective: Install and use Ruby gems for added functionality.
- Topics: Introduction to `Bundler`, Installing gems, Using a gem (`httparty` for making HTTP requests).
- Exercise:
    - Write a program that uses `httparty` to fetch data from an API (e.g., weather or a random quote) and display it.
    - Explore another gem (like `faker`) to generate random data for books in your `Library` app.

Day 2: Introduction to Gem Creation

- Objective: Learn how to structure and create a Ruby gem.
- Topics: Gem structure, Creating a `gemspec` file, Using `Bundler` to create a new gem.
- Exercise:
    - Create a small gem that prints a greeting message with customizable text (e.g., “Hello, [Name]!”).

Day 3: Writing Tests with RSpec

- Objective: Learn the basics of testing Ruby code with RSpec.
- Topics: Writing unit tests, Testing classes and methods, Testing edge cases.
- Exercise:
    - Write RSpec tests for the `Book` and `Library` classes.

    - Write tests for the small gem you created to ensure it handles different input cases correctly.

Day 4: Refactoring and Best Practices

- Objective: Focus on improving and cleaning up existing code using best practices.
- Topics: Refactoring techniques, Code readability, SOLID principles.
- Exercise:
    - Refactor the `Library` app to make the code more modular and readable.
    - Refactor the gem you created, ensuring it follows best practices like proper method naming and separation of concerns.

Day 5: Final Project - Building a Small App

- Objective: Consolidate all learning by building a simple app.
- Exercise:
    - Build a small console app that simulates a basic inventory system (e.g., a store that manages products). The app should allow adding products, listing products, updating product information, and managing stock levels.
    - Incorporate error handling and refactor the code based on feedback from Day 9's refactoring session.

NOTE: If you complete it within a day, that would be very appreciated, else please spend some time on the weekend and try to complete it.

Week 5: Rails Fundamentals & Application Structure

Day 1: Rails Project Setup and Application Structure

- **Objective:** Set up a new Rails application and understand its structure.
- **Topics:**
    - Rails directory structure
    - Generating models, controllers, and views
    - Routing basics
- **Exercise:**
    - Create a simple Rails app for managing posts (CRUD: Create, Read, Update, Delete).

Day 2: Active Record and Migrations

- **Objective:** Work with the database using Active Record and migrations.
- **Topics:**
  - Migrations (create, modify, delete tables)
  - Validations and database constraints
  - Active Record associations (has_many, belongs_to)
- **Exercise:**
  - Add a `Category` model for posts and create associations between `Post` and `Category`. Add validations to ensure each post has a title and a category.

Day 3: Controllers and Views

- **Objective:** Learn about controllers, views, and rendering templates.
- **Topics:**
  - Rendering partials
  - Controller actions and strong parameters
  - Forms and handling form submissions
- **Exercise:**
  - Create a form for creating and updating posts. Implement the ability to filter posts by category.

Day 4: Routes, RESTful APIs, and Resources

- **Objective:** Understand routing and RESTful design in Rails.
- **Topics:**
  - Resources routes
  - Nested routes
  - RESTful conventions (index, show, new, create, edit, update, destroy)
- **Exercise:**
  - Implement nested routes for `Categories` and `Posts`. Display posts belonging to a particular category in a nested view.

Day 5: Callbacks and Lifecycle

- **Objective:** Learn how Rails callbacks work and their use cases.
- **Topics:**
  - Before, after, and around callbacks
  - Best practices with callbacks
  - Model lifecycle hooks
- **Exercise:**

    - Add `before_save` and `after_create` callbacks to update a post's timestamp and send a notification after creation.

Week 6: Authentication, Authorization & Configuration

Day 1: Authentication with Devise

- **Objective:** Implement user authentication with the Devise gem.
- **Topics:**
    - Installing and configuring Devise
    - Setting up user login, registration, and session handling
    - Managing current user and session data
- **Exercise:**
    - Add user authentication to the posts app, allowing users to sign up, log in, and only view/edit their own posts.

Day 2: Authorization with CanCanCan

- **Objective:** Implement role-based authorization with CanCanCan.
- **Topics:**
    - Defining user roles and permissions
    - Restricting access to resources based on roles
- **Exercise:**
    - Set up roles (admin, editor, viewer) using CanCanCan. Allow admins to manage all posts, editors to manage their own, and viewers to only read posts.

Day 3: Role Management with Rolify

- **Objective:** Use Rolify for more granular role management.
- **Topics:**
    - Installing Rolify and configuring roles
    - Managing roles dynamically
- **Exercise:**
    - Add a role-based system to assign multiple roles to users (e.g., a user can be both an editor and a viewer).

Day 4: Rails Configuration (Environment Settings)

- **Objective:** Learn how to handle different environment configurations in Rails.

- **Topics:**
  - Configuration files (development, production, testing)
  - Using environment variables (dotenv-rails gem)
- **Exercise:**
  - Set up different environment configurations, using environment variables for sensitive data like API keys.

Day 5: Internationalization (i18n)

- **Objective:** Implement internationalization in the app.
- **Topics:**
  - Rails i18n API
  - Translating views, controllers, and models
- **Exercise:**
  - Translate the posts app into two languages (e.g., English and Spanish).

Week 7: Advanced Rails Features

Day 1: Associations and Nested Forms

- **Objective:** Learn about complex associations and nested forms.
- **Topics:**
  - Handling `has_many :through` and `has_and_belongs_to_many` associations
  - Creating nested forms with `fields_for`
- **Exercise:**
  - Allow users to tag posts with multiple categories using a `has_and_belongs_to_many` association.

Day 2: Active Storage

- **Objective:** Manage file uploads in Rails using Active Storage.
- **Topics:**
  - Configuring Active Storage with cloud providers (e.g., AWS S3)
  - Attaching files and displaying them
- **Exercise:**
  - Allow users to upload images for posts and display them on the post show page.

Day 3: Background Jobs with ActiveJob

- **Objective:** Implement background jobs using ActiveJob and Sidekiq.

- **Topics:**
  - Creating background jobs
  - Using Sidekiq for background processing
- **Exercise:**
  - Send an email notification (using ActionMailer) after a post is created, processed in the background with Sidekiq.

Day 4: ActionCable (WebSockets)

- **Objective:** Implement real-time features using ActionCable.
- **Topics:**
  - Setting up ActionCable
  - Real-time updates in Rails apps
- **Exercise:**
  - Implement live comments on posts using ActionCable.

Day 5: API Development (JSON and REST)

- **Objective:** Build a JSON API for the app.
- **Topics:**
  - Rendering JSON responses
  - Versioning APIs
  - API authentication (Token-based auth)
- **Exercise:**
  - Create a JSON API for managing posts, allowing external clients to create and retrieve posts.

Week 8: Testing, Debugging, and Final Project

Day 1: Testing with RSpec and FactoryBot

- **Objective:** Write tests for your Rails application.
- **Topics:**
  - Setting up RSpec for testing
  - Using FactoryBot for test data
  - Writing controller, model, and integration tests
- **Exercise:**
  - Write tests for user authentication and post creation.

Day 2: Debugging and Profiling

- **Objective:** Learn debugging techniques in Rails.
- **Topics:**
    - Using `byebug` for debugging
    - Profiling code for performance bottlenecks
- **Exercise:**
    - Debug a bug in the post creation flow and profile the app for performance issues.

Day 3: Caching in Rails

- **Objective:** Implement caching for performance optimization.
- **Topics:**
    - Fragment caching
    - Russian doll caching
- **Exercise:**
    - Cache post views to optimize page load time, using fragment caching.

Day 4: Final Project - Part 1

- **Objective:** Start building a final project using all learned concepts.
- **Exercise:**
    - Build a multi-user blog platform with authentication, authorization, image uploads, and live comments using ActionCable.

Day 5: Final Project - Part 2

- **Objective:** Complete and refine the final project.
- **Exercise:**
    - Finish and polish the blog platform. Add testing, background jobs, and caching to enhance the app.

Week 9: Introduction to Stimulus.js and Basic Concepts

Day 1: Introduction to Stimulus.js

- **Objective:** Understand what Stimulus.js is and how it integrates with Rails.
- **Topics:**
    - Overview of Stimulus.js
    - Why use Stimulus.js with Rails
    - Setting up Stimulus in a Rails project
- **Exercise:**

    - Set up a basic Rails app with Stimulus.js and create a simple Stimulus controller to handle a button click event.

Day 2: Stimulus Controllers

- **Objective:** Learn the basics of Stimulus controllers and actions.
- **Topics:**
    - Stimulus controller structure
    - Defining and triggering actions
    - Connecting controllers to HTML elements
- **Exercise:**
    - Create a Stimulus controller to toggle visibility of a paragraph when a button is clicked.

Day 3: Targets and Data Attributes

- **Objective:** Use Stimulus targets and data attributes for DOM manipulation.
- **Topics:**
    - Working with targets in Stimulus
    - Accessing and modifying DOM elements through data attributes
- **Exercise:**
    - Create a Stimulus controller that displays dynamic content based on button clicks using targets.

Day 4: Stimulus Values

- **Objective:** Learn to use values in Stimulus controllers.
- **Topics:**
    - Defining and using values in Stimulus
    - Updating values dynamically
- **Exercise:**
    - Build a simple counter using Stimulus values, allowing the user to increment and decrement the count.

Day 5: Lifecycle Methods and Controller Lifecycle

- **Objective:** Understand Stimulus controller lifecycle callbacks.
- **Topics:**
    - Lifecycle methods: `connect`, `disconnect`, `initialize`
    - Using lifecycle methods to manage state
- **Exercise:**

    - Create a Stimulus controller that initializes with a certain value and updates the UI when connected/disconnected.

Week 10: Advanced Stimulus.js Features and Practical Applications

Day 1: Working with External Libraries

- **Objective:** Learn how to integrate external libraries with Stimulus.js.
- **Topics:**
    - Using Stimulus with external JavaScript libraries (e.g., date pickers, charts)
    - Managing third-party integrations within Stimulus controllers
- **Exercise:**
    - Integrate a date picker library with Stimulus and update a form input field dynamically when a date is selected.

Day 2: Handling Forms with Stimulus.js

- **Objective:** Enhance form handling and validations using Stimulus.
- **Topics:**
    - Listening to form events (submit, change)
    - Client-side form validation with Stimulus
- **Exercise:**
    - Create a form with dynamic validation feedback using Stimulus, showing error messages before submission.

Day 3: AJAX and Fetch API with Stimulus

- **Objective:** Use Stimulus to handle AJAX requests and interact with the backend.
- **Topics:**
    - Sending AJAX requests using the Fetch API
    - Handling responses and updating the DOM
- **Exercise:**
    - Build a Stimulus controller that fetches data from an API and updates the content of a page without reloading.

Day 4: Stimulus Reflex and CableReady

- **Objective:** Explore integrating Stimulus with real-time features.
- **Topics:**
    - Introduction to Stimulus Reflex and CableReady

    - Building real-time UI components
- **Exercise:**
    - Implement a basic real-time notifications system with Stimulus Reflex, updating the UI when new data arrives.

Day 5: Final Stimulus.js Project

- **Objective:** Combine all the learned concepts to create a practical application using Stimulus.js.
- **Exercise:**
    - Create a multi-step form using Stimulus controllers, handling each step dynamically with validations, and sending the final form data using AJAX.

Week 11: Turbo with Stimulus

Day 1: Introduction to Turbo and Turbo Drive

- **Objective:** Understand the basics of Turbo and how it enhances the Rails app experience.
- **Topics:**
    - Overview of Turbo in Hotwire
    - What is Turbo Drive? (Faster navigation without full-page reloads)
    - Enabling Turbo Drive in a Rails app
- **Exercise:**
    - Create a Rails app with Turbo Drive enabled. Use Turbo to speed up navigation between pages by replacing traditional links with Turbo-driven links.

Day 2: Turbo Frames

- **Objective:** Learn to use Turbo Frames for partial page updates.
- **Topics:**
    - Introduction to Turbo Frames
    - How Turbo Frames reduce unnecessary page reloads
    - Using `<turbo-frame>` to update only sections of a page
- **Exercise:**
    - Implement Turbo Frames to update a section of a page (e.g., a comments section that loads new comments without reloading the entire page).

Day 3: Combining Turbo Frames and Stimulus

- **Objective:** Use Turbo Frames with Stimulus for dynamic interaction.

- **Topics:**
    - Combining Stimulus.js controllers with Turbo Frames for reactive elements
    - Managing state changes in Turbo Frames with Stimulus
- **Exercise:**
    - Create a Stimulus controller that listens to changes in a Turbo Frame and updates the page dynamically (e.g., update the UI when new data is loaded into a Turbo Frame).

Day 4: Turbo Streams

- **Objective:** Use Turbo Streams to handle real-time updates on the client-side.
- **Topics:**
    - Introduction to Turbo Streams
    - Sending updates via Turbo Streams to modify the DOM
    - Using Turbo Streams with Rails models and broadcasting changes
- **Exercise:**
    - Implement Turbo Streams to add new comments to a blog post in real time without refreshing the page.

Day 5: Advanced Turbo Stream Features

- **Objective:** Dive deeper into more advanced Turbo Stream functionality.
- **Topics:**
    - Customizing Turbo Stream actions
    - Using Turbo Stream templates
    - Broadcasting updates from models using Turbo Streams
- **Exercise:**
    - Create an application that broadcasts changes (e.g., adding, updating, or deleting records) to all connected clients in real-time using Turbo Streams.

BEST OF LUCK!