BikeBecho.com

From Concept to Code: A Full Walkthrough

Presentation Agenda

Part 1: Project Overview

- Project Goals & Concept
- Visual Identity & Design System
- User Flow & Site Architecture

Part 2: Technical Deep Dive

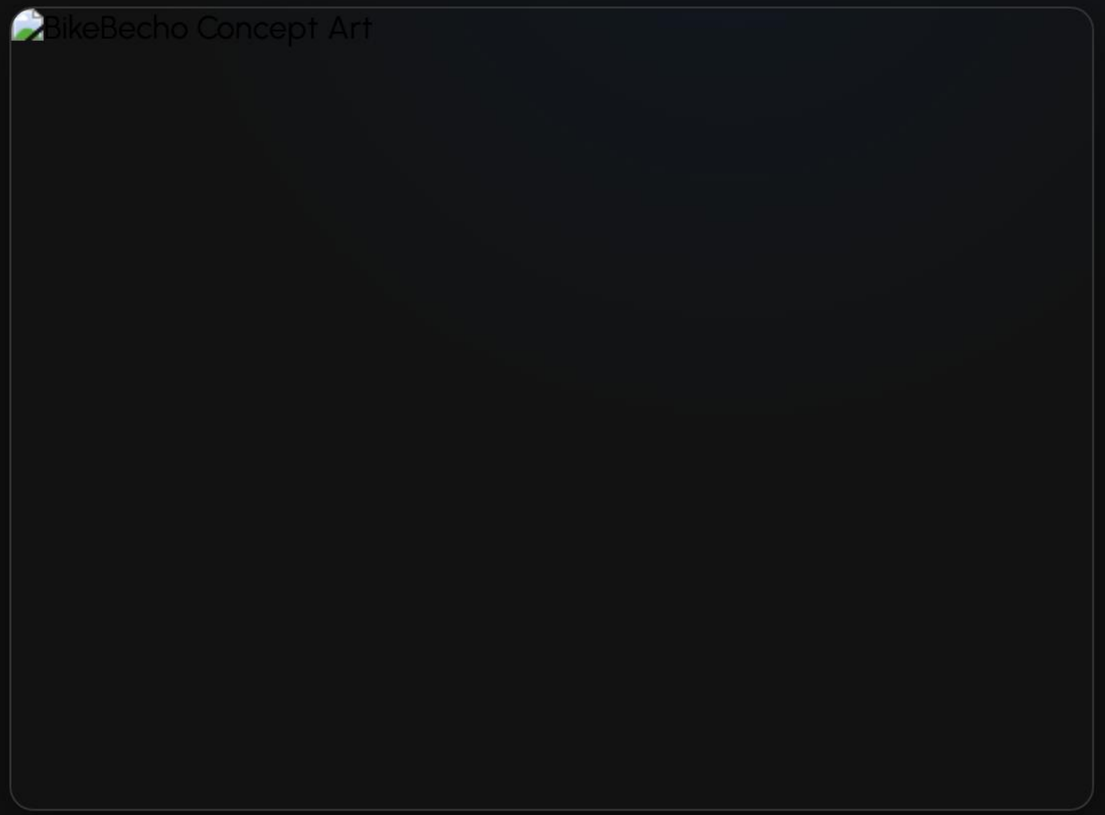
- **Architecture:** File Structure
- **HTML5:** Semantic Structure & Full Code
- **CSS3:** Styling, Responsiveness & Full Code
- **JavaScript:** Validation Logic & Full Code

The Concept

What is BikeBecho?

BikeBecho is a prototype web platform designed to streamline the process of buying and selling used two-wheelers. It bridges the gap between sellers and potential buyers with a clean, transparent digital interface.

Primary Goal: To demonstrate a robust front-end architecture using core web technologies without relying on frameworks.



Technology Stack

HTML5

Provides the semantic structure, accessibility features, and content organization for all pages.

CSS3

Handles the visual presentation, responsive layouts, color themes, and interactive hover states.

JavaScript

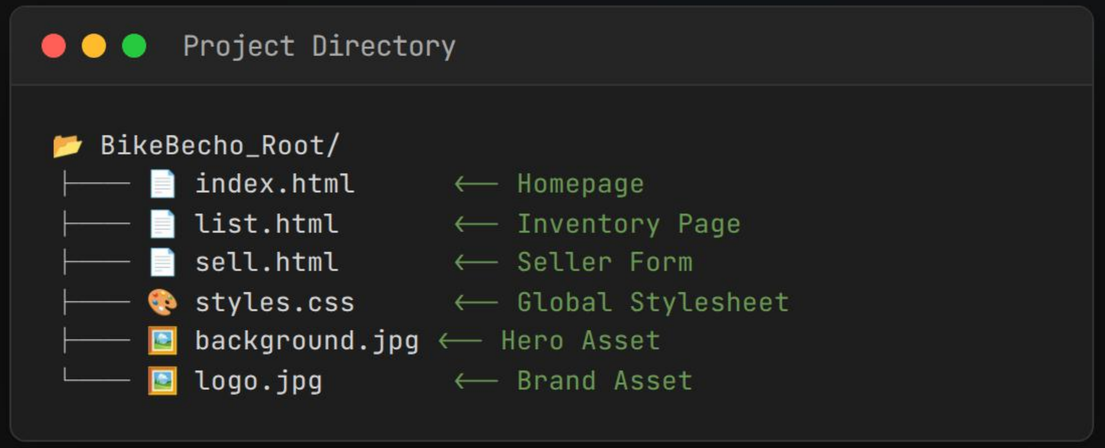
Powers the client-side logic, specifically handling form submission events and data validation.

1. File Architecture

Modular Organization

The project follows a "Separation of Concerns" methodology. Instead of cluttering HTML with styles and scripts, we maintain distinct files.

- **Maintainability:** Easy to update styles globally.
- **Performance:** Browser caches the CSS file.
- **Readability:** Clean code structure.



2. HTML Structure (Homepage) - Deep Dive

Semantic Markup Strategy

The `index.html` serves as the entry point. Key strategies used:

- : Connects the external CSS file, ensuring a clean separation of concerns.
- : Contains the Logo and Navigation for consistent site structure.
- `.hero`: A dedicated section for the immersive background image and main call-to-action.

↓ Full Code on Next Slide

index.html (Snippet)

```
<head>
  <link rel="stylesheet" href="styles.css">
head>
<body>
  <section class="hero">
    <h2>Your Ride, Your Ruleshh2>
  section>
```

Full Source Code: index.html

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>BikeBecho - Buy & Sell Used Bikestitle>
  <link rel="stylesheet" href="styles.css">
head>
<body>

  <header>
    <a href="index.html">
      <img src="logo.jpg" alt="BikeBecho Logo" class="logo">
    a>
    <nav>
      <ul>
        <li><a href="index.html">Homea>li>
        <li><a href="list.html">View Listed Bikesa>li>
        <li><a href="sell.html">Sell Your Bikea>li>
        <li><a href="#about">About Usa>li>
        <li><a href="#contact">Contact Usa>li>
      ul>
    nav>
  header>
```

3. Styling & Theming - Deep Dive

Visual Mechanics

We use CSS to manage the background image effectively for visual appeal and critical text readability.

- `background-size: cover` ensures the image fills the screen responsively.
- The `::before` pseudo-element creates a **dark overlay** (40% opacity). This ensures white text is readable against a busy image background.

↓ Full Code on Next Slide

styles.css (Snippet)

```css
.hero {
  background: url('background.jpg') center/cover;
  position: relative;
}

.hero::before {
  content: '';
  background: rgba(0, 0, 0, 0.4);
}
```

Full Source Code: styles.css

styles.css

```css
/* Base Styles */
body { font-family: Arial, sans-serif; margin: 0; padding: 0; background-color: #f4f4f4; color: #333; line-height: 1.6; }

/* Header */
header { background-color: #007bff; padding: 10px 0; text-align: center; box-shadow: 0 2px 5px rgba(0,0,0,0.2); }
header .logo { max-width: 250px; height: auto; margin-bottom: 10px; }

/* Navigation */
nav ul { list-style-type: none; padding: 0; margin: 0; text-align: center; }
nav ul li { display: inline-block; margin: 0 15px; }
nav ul li a { color: white; text-decoration: none; font-weight: bold; padding: 8px 15px; border-radius: 5px; transition: background-color
0.3s ease; display: block; }
nav ul li a:hover { background-color: #0056b3; }

/* Hero Section */
.hero {
  background: url('background.jpg') no-repeat center center/cover;
  color: white; text-align: center; padding: 100px 20px; min-height: 400px;
  position: relative; box-shadow: inset 0 -5px 10px rgba(0,0,0,0.3);
}
.hero::before {
  content: ''; position: absolute; top: 0; left: 0; right: 0; bottom: 0;
  background: rgba(0, 0, 0, 0.4); z-index: 1;
}
```

4. Listing Layout (list.html) - Deep Dive

list.html (Snippet)

```
<div class="bike-card">
  <h3>Royal Enfield Classich3>
  <p><strong>Price:strong> ₹1.2Lp>
  <p><strong>Loc:strong> Bangalorep>
div>
```

The Card Component

We created a reusable `.bike-card` class. This modular approach allows us to display 10 or 100 bikes with consistent styling.

- **Scalability:** Change CSS once, update all cards.
- **Shadows:** `box-shadow` adds depth to separate cards from the background.
- **Accent Border:** A left-border color draws attention.

↓ Full Code on Next Slide

Full Source Code: list.html

list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Bike Listings - BikeBechotitle>
  <link rel="stylesheet" href="styles.css">
head>
<body>

  <header>
    <h1>View Listed Bikesh1>
    <nav>
      <ul>
        <li><a href="index.html">Homea>li>
        <li><a href="list.html">View Listed Bikesa>li>
        <li><a href="sell.html">Sell Your Bikea>li>
      ul>
    nav>
  header>

  <div class="container">
    <h2>Available Two-Wheelers for Saleh2>
```

5. The Seller Form (sell.html) - Deep Dive

Data Input & Native Validation

The form is the interactivity hub. We use specific HTML5 input types to ensure data quality right from the UI level.

- `id="sellBikeForm"`: The primary hook for all JavaScript interaction.
- `type="number"`: Prevents non-numeric input for Price/Year.
- `required` and `min/max`: Browser-native constraint validation, providing immediate user feedback.

↓ Full Code on Next Slide

sell.html (Snippet)

```
<form id="sellBikeForm">
  <label>Model:label>
  <input type="text" required>

  <label>Price (₹):label>
  <input type="number" id="price" min="5000">

  <input type="submit" value="List">
form>
```

Full Source Code: sell.html

sell.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Sell Your Bike - BikeBechotitle>
  <link rel="stylesheet" href="styles.css">
head>
<body>
  <header>
    <h1>Register Your Bike to Sellh1>
    <nav>
      <ul>
        <li><a href="index.html">Homea>li>
        <li><a href="list.html">View Listed Bikesa>li>
        <li><a href="sell.html">Sell Your Bikea>li>
      ul>
    nav>
  header>

  <div class="container">
    <div class="form-content">
      <h2>Fill in the details to list your bikeh2>
      <form id="sellBikeForm" action="#" method="POST">
        <label for="make">Bike Make:label>
```

6. JavaScript Interactivity - Deep Dive

Logic Breakdown

We add interactivity to the form using a simple script at the bottom of `sell.html`.

- **Event Listener:** Listens for the `submit` event on the form.
- **e.preventDefault():** Stops the page reload so we can handle data via JS.
- **Validation Logic:** Checks if Year > 2025 or Price < 1000.
- **Feedback:** Displays dynamic success/error messages.

↓ Full Code on Next Slide

Script Snippet

```
document.getElementById('sellBikeForm')
  .addEventListener('submit', function(e) {
  e.preventDefault();
  // ... Validation Logic ...
});
```

Full Source Code: JavaScript Block

sell.html (Script Tag)

```
</div>
document.getElementById('sellBikeForm').addEventListener('submit', function(event) {
  // 1. Prevent Default Submission
  event.preventDefault();

  // 2. Get Form Elements
  const form = event.target;
  const messageBox = document.getElementById('formMessage');
  let isValid = true;
  let errorMessage = '';

  // 3. Parse and Validate Values
  const year = parseInt(document.getElementById('year').value);
  const mileage = parseInt(document.getElementById('mileage').value);
  const price = parseInt(document.getElementById('price').value);

  if (year < 1990 || year > 2025) {
    isValid = false;
    errorMessage = 'Invalid Year (1990-2025).';
  } else if (mileage < 0) {
    isValid = false;
    errorMessage = 'Mileage cannot be negative.';
  } else if (price < 1000) {
    isValid = false;
```

Future Roadmap

1. **Backend**
   Node.js Server
2. **Database**
   MongoDB Storage
3. **Auth**
   User Login
4. **Payments**
   Secure Checkout
5. **Live**
   Cloud Deploy

Thank You!

The BikeBecho project demonstrates the power of fundamental web technologies.

Questions?

github.com/bikebecho