Flutter Forms & Validation: Building User Input Forms

Subtitle: Learn how to create forms, validate input, and handle user data in Flutter apps.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Forms are essential for **collecting user input**. Flutter provides tools like `Form`, `TextFormField`, and validators to **ensure correct data** before submission. This guide covers practical form building and validation.

Step 1: Create a Basic Form

```
final _formKey = GlobalKey<FormState>();

Form(
  key: _formKey,
  child: Column(
    children: [
      TextFormField(
        decoration: InputDecoration(labelText: 'Email'),
      ),
      ElevatedButton(
        onPressed: () {
          if (_formKey.currentState!.validate()) {
            print('Form is valid');
          }
        },
        child: Text('Submit'),
      ),
    ],
  ),
)
```

Exercise: Build a simple form with an email field and a submit button.

Step 2: Add Validators

```
TextFormField(
  decoration: InputDecoration(labelText: 'Email'),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter email';
    }
    if (!value.contains('@')) {
      return 'Enter a valid email';
    }
    return null;
  },
)
```

Exercise: Add validation for empty fields and proper email format.

Step 3: Password Field with Validation

```
TextFormField(
  obscureText: true,
  decoration: InputDecoration(labelText: 'Password'),
  validator: (value) {
    if (value == null || value.length < 6) {
      return 'Password must be at least 6 characters';
    }
    return null;
  },
)
```

Exercise: Add a password field with a minimum character requirement.

Step 4: Collect Form Data

```
String email = '';
String password = '';

TextFormField(
  decoration: InputDecoration(labelText: 'Email'),
  onSaved: (value) => email = value!,
)
```

- Use `_formKey.currentState!.save()` after validation to save input values

Exercise: Print email and password after form submission.

Step 5: Mini Project

Build a **Flutter Login Form**:

- Email and password fields with validation
- Submit button to print collected values
- Show error messages for invalid input
- Bonus: Add a "Confirm Password" field for signup forms

Key Takeaways

- `Form` and `TextFormField` handle user input efficiently
- Validators ensure **correct data before submission**
- `onSaved` collects data after validation
- Proper forms improve **UX and data integrity**
- Forms are essential for **login, signup, and data collection apps**

Skipping form validation leads to **bad user experience and potential data errors**, which can break app functionality.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
