Git Flow + Validation

Asst.Prof.Drusawin Vongpramate

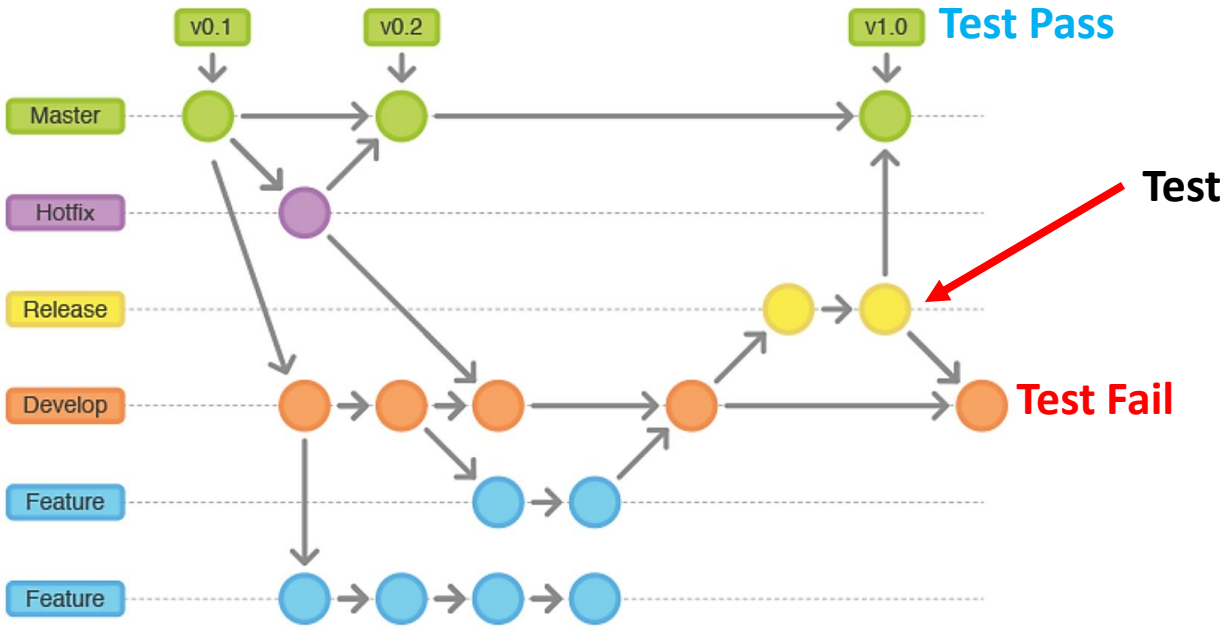
Department of Information Technology

Faculty of Science, BRU

Hint

Main Topic Sub Topic

Git Flow



> git branch develop //create branch

> git checkout develop //use branch

Git Flow

hotfix

> git branch hotfix/loginError

release

> git checkout develop

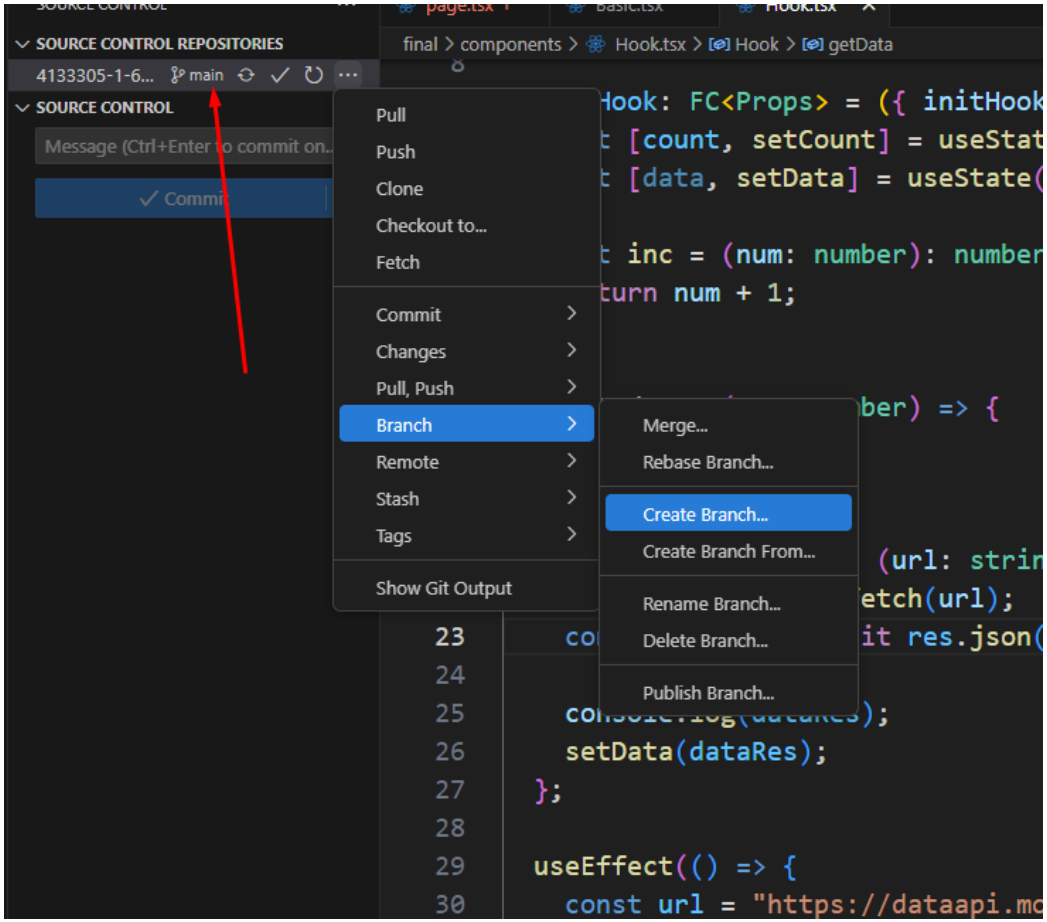
> git branch release/0.0.1

Feature

> git checkout develop

> git branch feature/profile_management

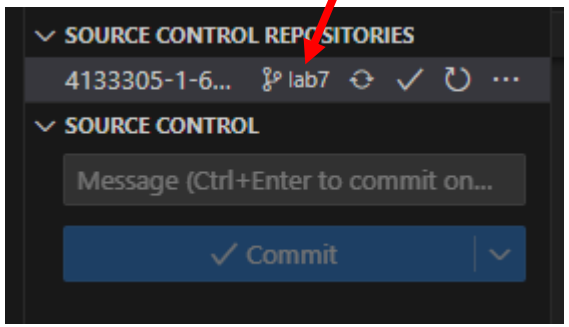
Create Git branch



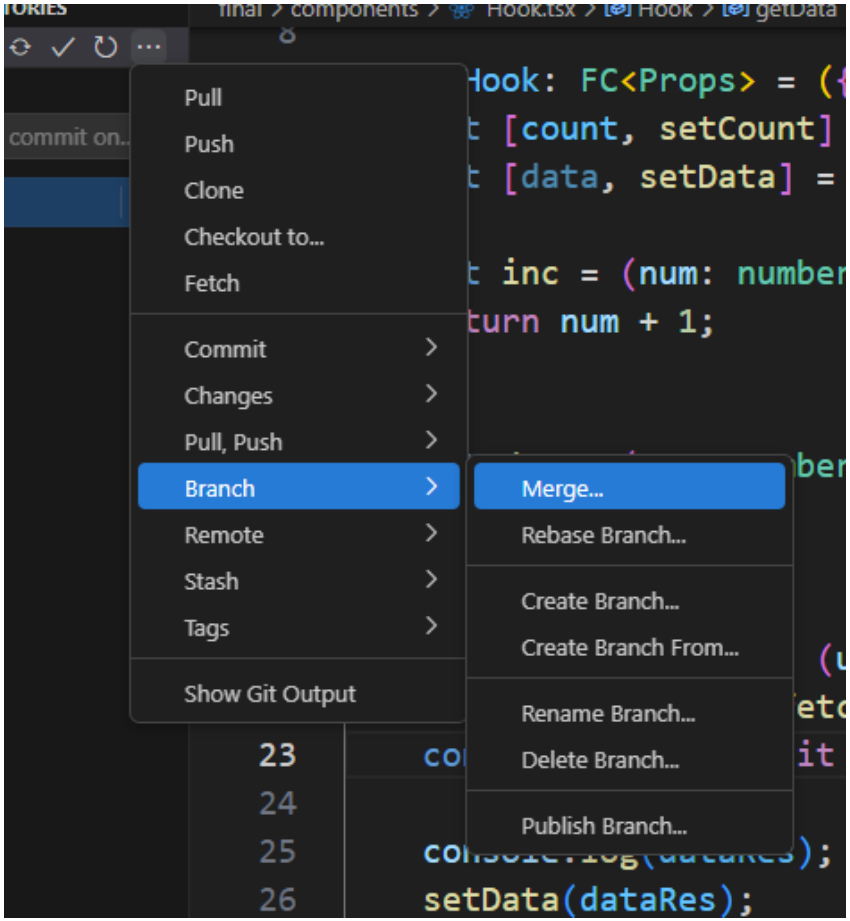

```
C:\Users\HoMeW\Desktop\4133305-1-67>git branch
* lab7
  main
```

> git checkout lab7

current working branch is “**lab7**”

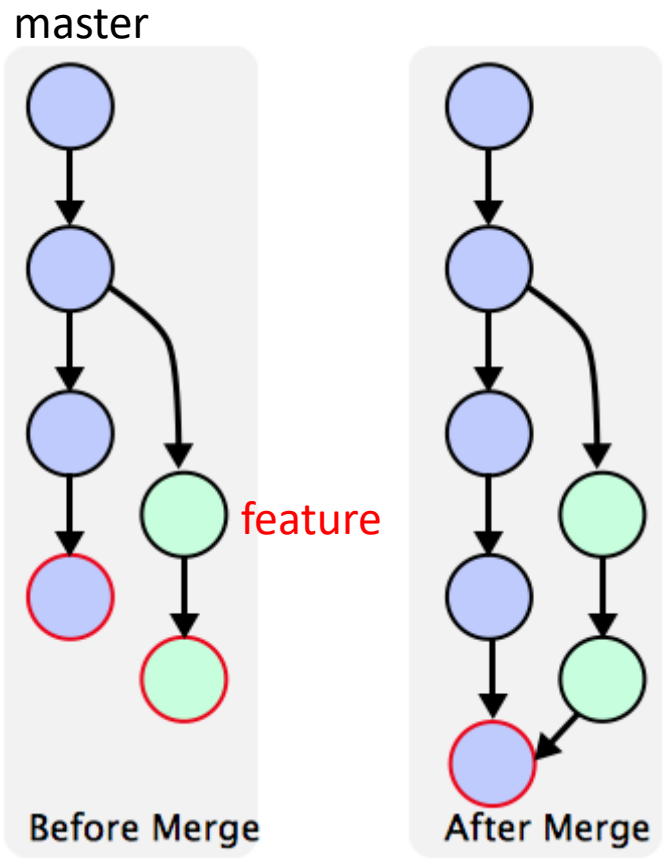


Merge Git branch



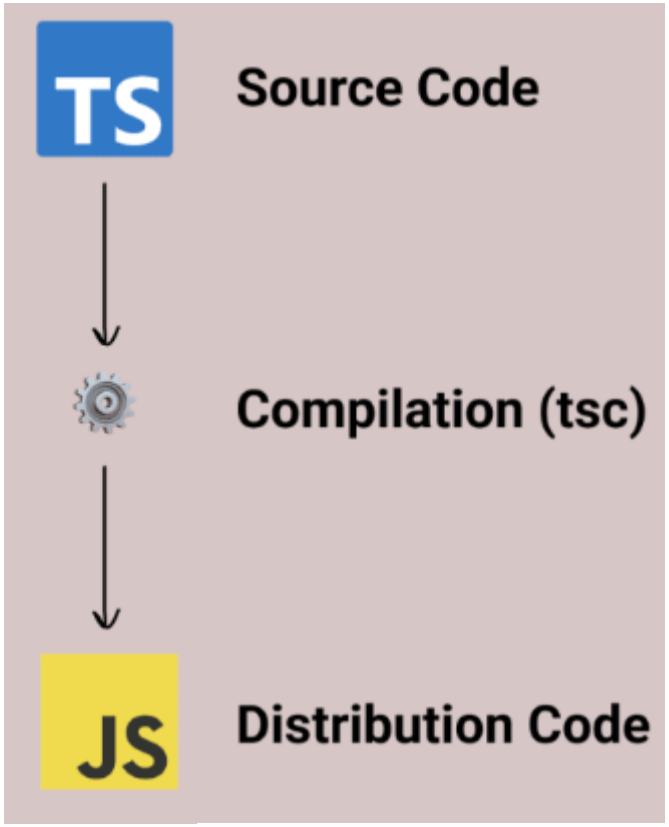
Merge to working branch

> git merge feature/profile_management

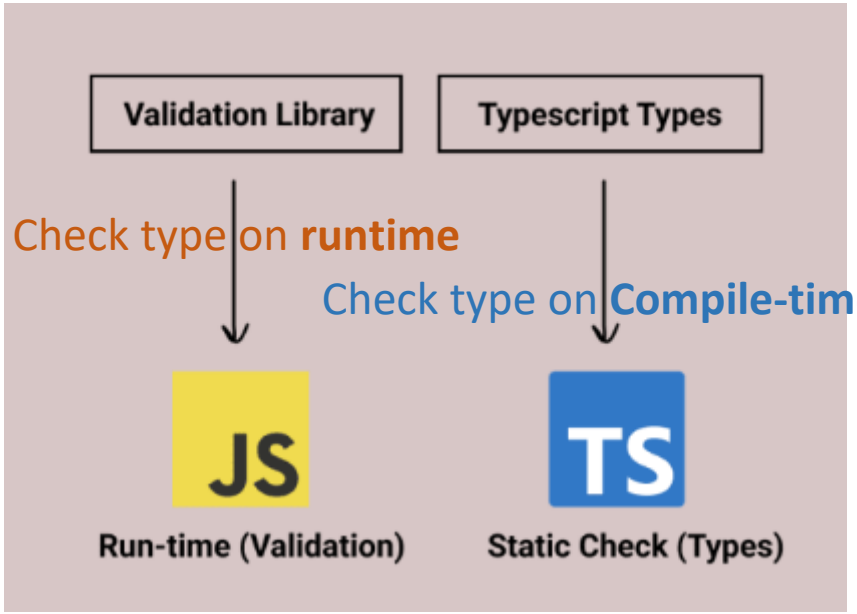


Zod

Understanding

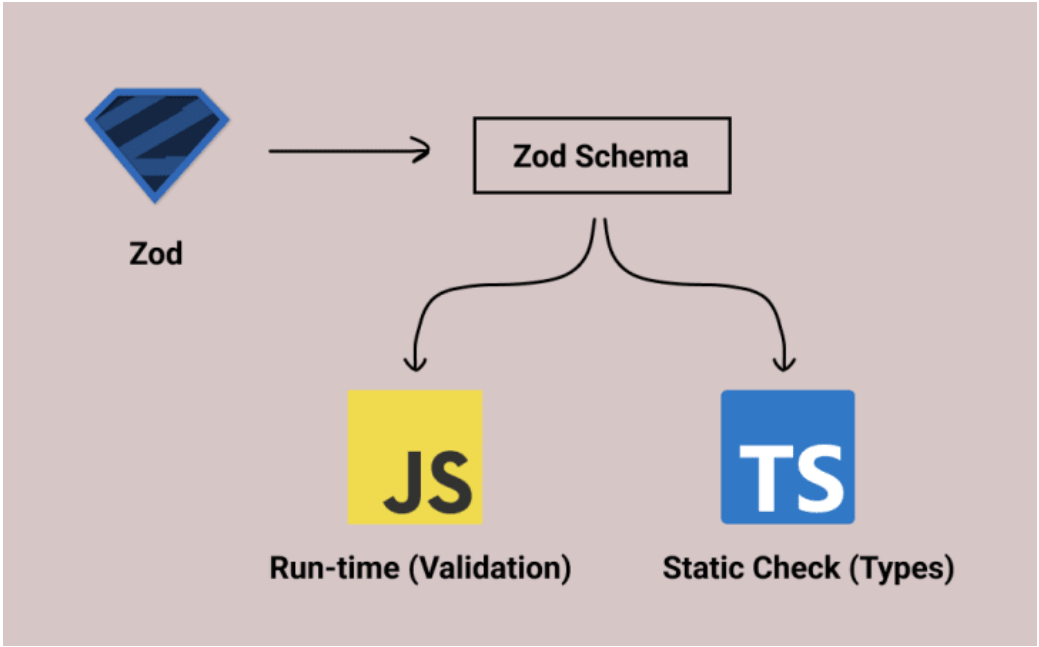


Traditional approach



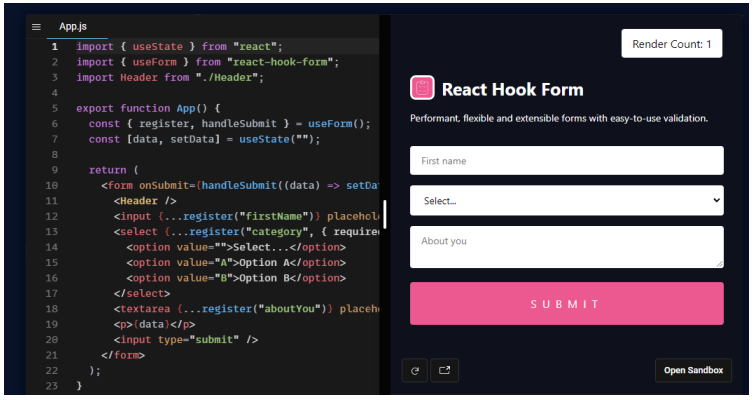
Zod

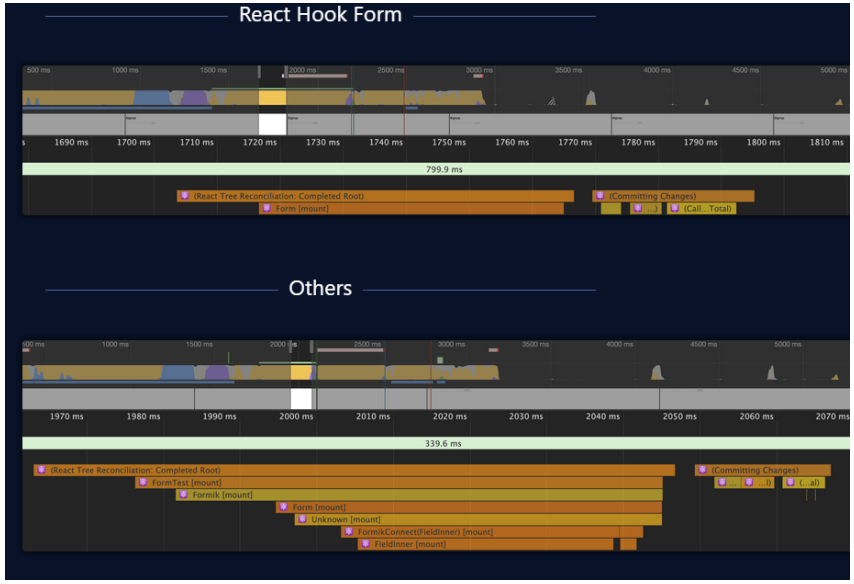
Zod is a TypeScript-first **schema declaration** and **validation** library. The term "schema" is broadly used to refer to any data structure, ranging from a basic string to a complex, nested object.



react-hook-form

Performant, flexible and extensible forms with easy-to-use validation. React Hook Form reduces the amount of code you need to write while removing unnecessary re-renders.



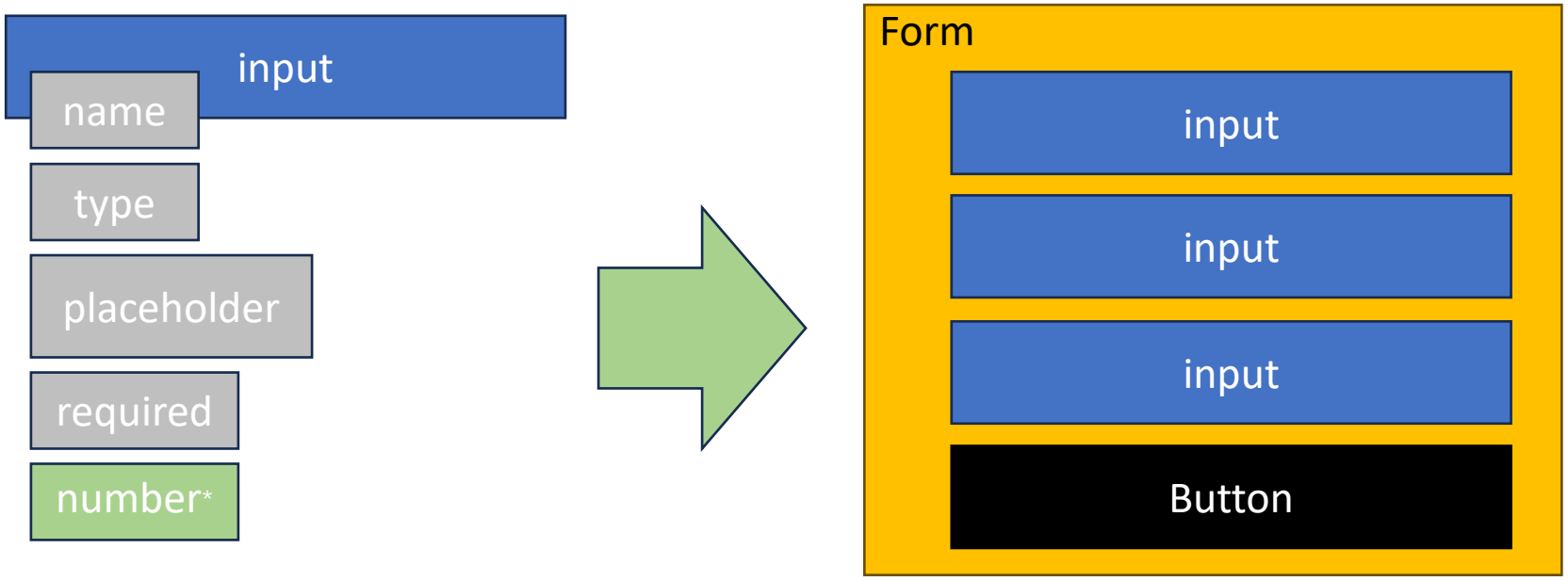


Example

> npm i react-hook-form @hookform zod -s

Step

1. Design Component
2. Define type
3. Create Component & Form
4. Call



final > components > form > types.tsx > ...

```tsx
import { FieldError, UseFormRegister } from "react-hook-form";

export type ValidFieldNames =
  | "email"
  | "githubUrl"
  | "yearsOfExperience"
  | "password"
  | "confirmPassword";

export type FormData = {
  email: string;
  githubUrl: string;
  yearsOfExperience: number;
  password: string;
  confirmPassword: string;
};

export type FormFieldProps = {
  register: UseFormRegister<FormData>;
  name: ValidFieldNames;
  error: FieldError | undefined;
  type: string;
  placeholder: string;
  valueAsNumber?: boolean;
  required?: boolean;
};
```

final > components > form > FormField.tsx > ...

```tsx
import { FormFieldProps } from "./types";

const FormField: React.FC<FormFieldProps> = ({
  type,
  placeholder,
  name,
  register,
  error,
  valueAsNumber,
  required,
}) => (
  <>
    <input
      type={type}
      placeholder={placeholder}
      required={required}
      {...register(name, { valueAsNumber })}
    />
    {error && <span>{error.message}</span>}
  </>
);

FormField.defaultProps = {
  required: false,
};

export default FormField;

```

Input provided props (type, placeholder, name, required).

register the input field with the form, enabling form state management.

store as number (option)

final > components > form > Form.tsx > ...

```tsx
import { useForm } from "react-hook-form";
import { FormData } from "./types";
import FormField from "./FormField";

function Form() {
  const {
    register,
    handleSubmit,
    formState: { errors },
    setError,
  } = useForm<FormData>();

  const onSubmit = async (data: FormData) => {
    console.log("SUCCESS", data);
  };

  return (
    <form onSubmit={handleSubmit(onSubmit)}>
      <div>
        <h1>Zod & React-Hook-Form</h1>
        <FormField
          type="email"
          placeholder="Email"
          name="email"
          register={register}
          required
          error={errors.email}
        />

        <FormField
          type="text"
          placeholder="GitHub URL"
          name="githubUrl"
          register={register}
          error={errors.githubUrl}
        />

        <FormField
          type="number"
          placeholder="Years of Experience (1 - 10)"
          name="yearsOfExperience"
          register={register}
          error={errors.yearsOfExperience}
          valueAsNumber
        />

        <FormField
          type="password"
          placeholder="Password"
          name="password"
          register={register}
          error={errors.password}
        />

        <FormField
          type="password"
          placeholder="Confirm Password"
          name="confirmPassword"
          register={register}
          error={errors.confirmPassword}
        />
        <button type="submit" className="submit-button">
          Submit
        </button>
      </div>
    </form>
  );
}

export default Form;
```

http://127.0.0.1:3000/login

final > app > login > page.tsx > ...

```tsx
"use client";
import Form from "@/components/form/Form";

function Login() {
  return (
    <>
      <Form />
    </>
  );
}

export default Login;
```

Define a Form Schema with Zod

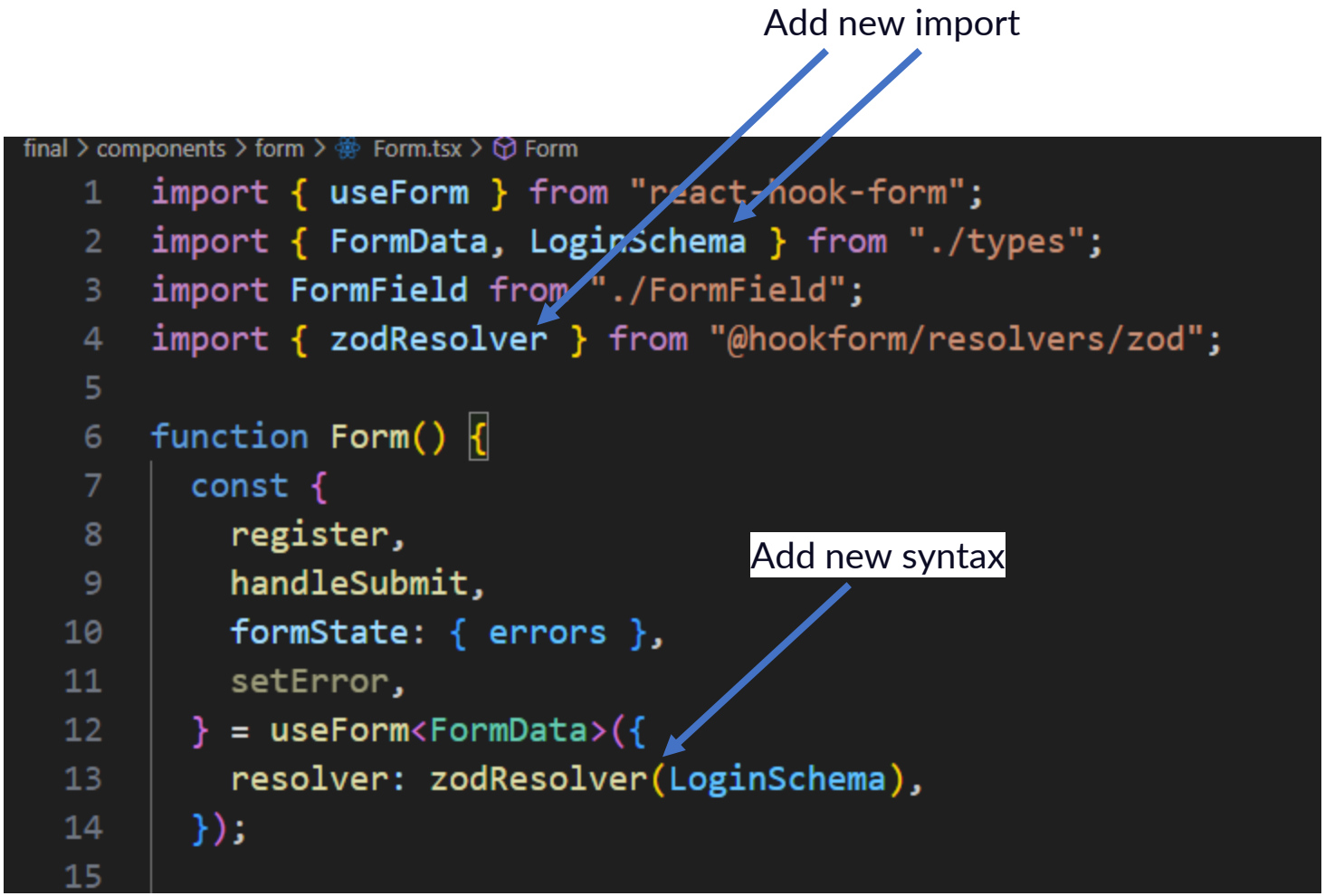
final > components > form > types.tsx > FormData

```tsx
import { FieldError, UseFormRegister } from "react-hook-form";
import { z, ZodType } from "zod";
```

Add new import

```tsx
export const LoginSchema: ZodType<FormData> = z
  .object({
    email: z.string().email(),
    githubUrl: z
      .string()
      .url()
      .includes("github.com", { message: "Invalid GitHub URL" }),
    yearsOfExperience: z
      .number({
        required_error: "required field",
        invalid_type_error: "Years of Experience is required",
      })
      .min(1)
      .max(10),
    password: z
      .string()
      .min(8, { message: "Password is too short" })
      .max(20, { message: "Password is too long" }),
    confirmPassword: z.string(),
  })
  .refine((data) => data.password === data.confirmPassword, {
    message: "Passwords do not match",
    path: ["confirmPassword"], // path of error
  });
```

Integrate Zod with React-Hook-Form for validation

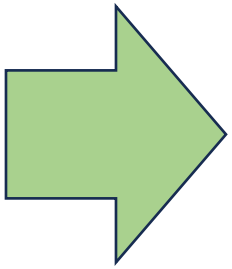


http://127.0.0.1:3000/login

Zod & React-Hook-Form

bfxv@d.c	Invalid email
http://github.co	Invalid GitHub URL
Years of Experience (1 - 1	Years of Experience is required
••••••	Password is too short
•••••••	Passwords do not match

Submit



Zod & React-Hook-Form

bfxv@d.cc	
http://github.com/	
2	
••••••••	
••••••••	Submit

Console

```
SUCCESS                                                                    Form.tsx:17
▼ {email: 'bfxv@d.cc', githubUrl: 'http://github.com/', yearsOfExperience: 2, password: '11111111', confirmPassword: '1111111
  1'} ℹ
    confirmPassword: "11111111"
    email: "bfxv@d.cc"
    githubUrl: "http://github.com/"
    password: "11111111"
    yearsOfExperience: 2
  ▶ [[Prototype]]: Object
>
```

Validate data from the **Register form.**

Guideline

1. What is schema?
2. What is components?
3. How to handle input issue and error?

Q & A