

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
```

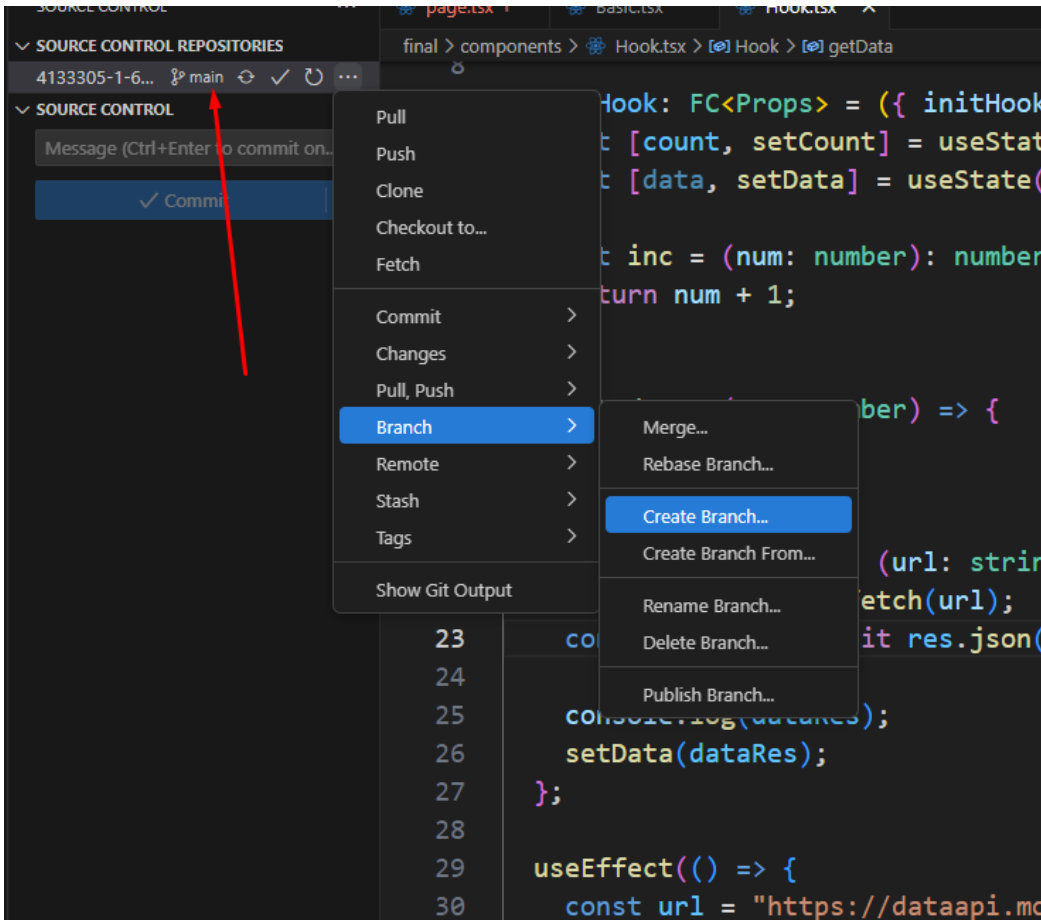
```
> git branch release/0.0.1
```

Feature

```
> git checkout develop
```

```
> git branch feature/profile_management
```

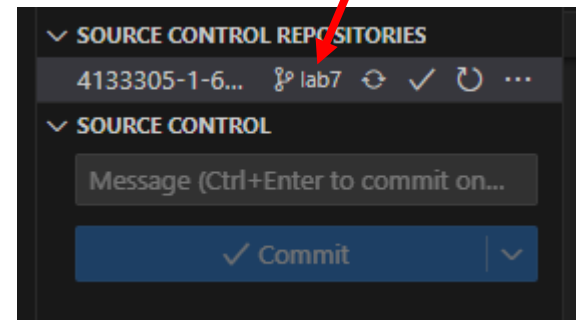
Create Git branch



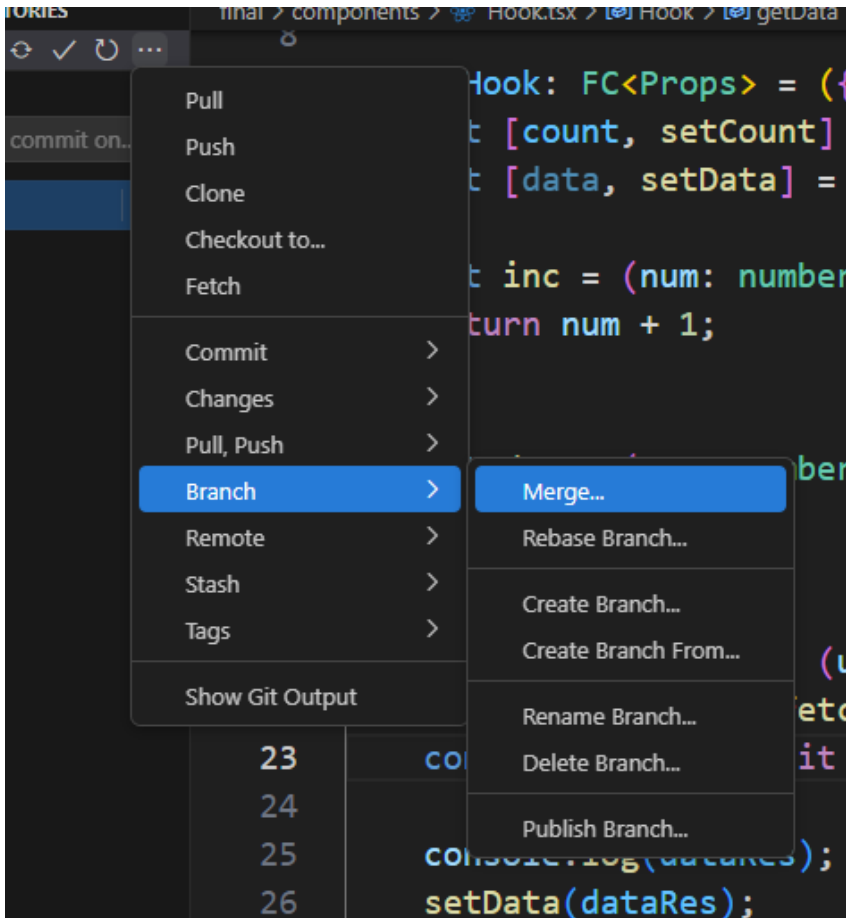
```
C:\Users\HoMew\Desktop\4133305-1-67>git branch
* lab7
  main
```

```
> git checkout lab7
```

current working branch is "lab7"



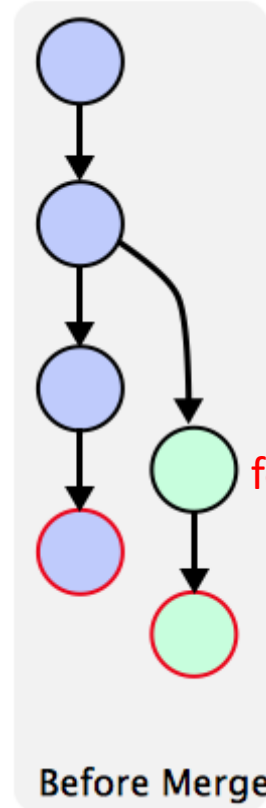
Merge Git branch



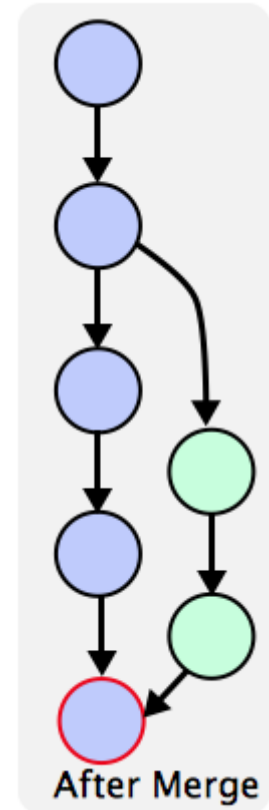
Merge to working branch

```
> git merge feature/profile_management
```

master

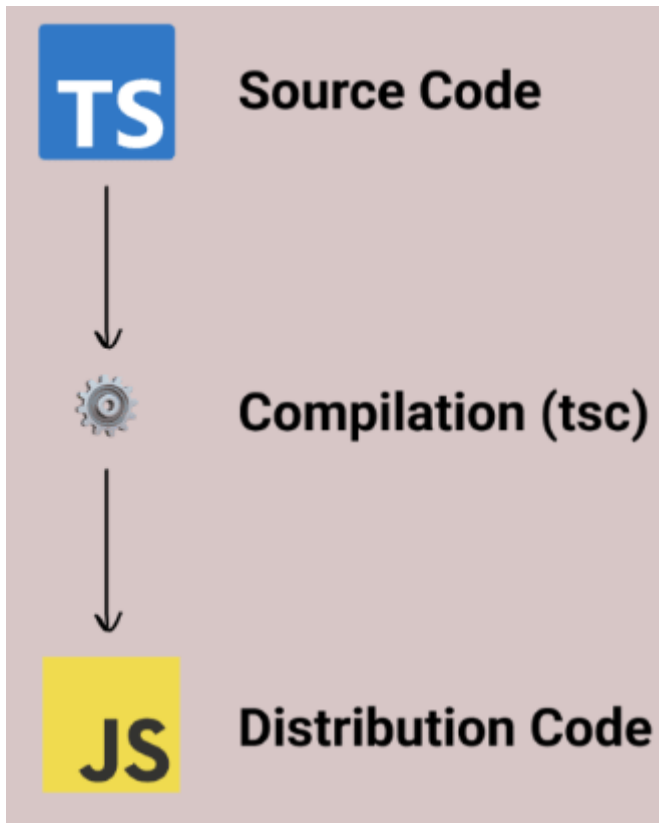


feature

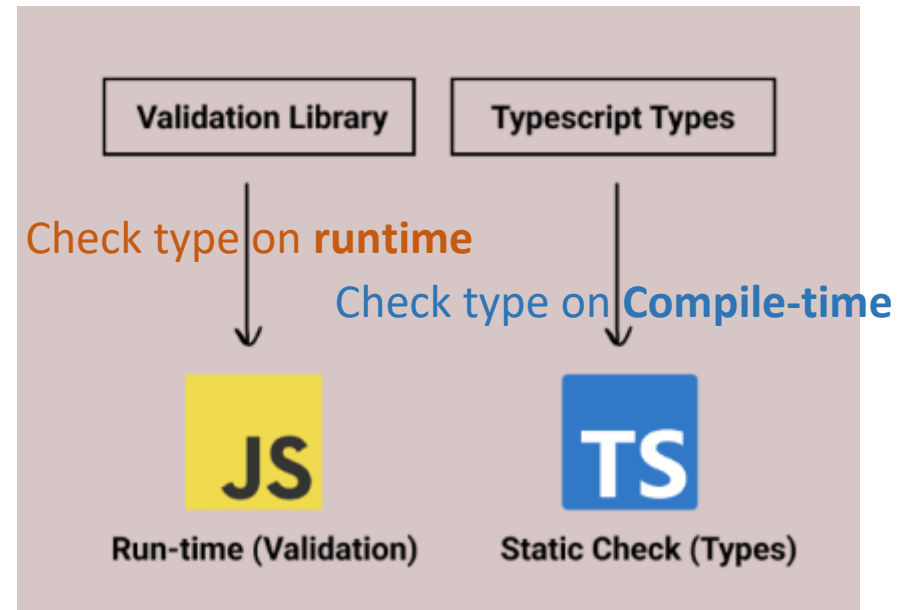


Zod

Understanding

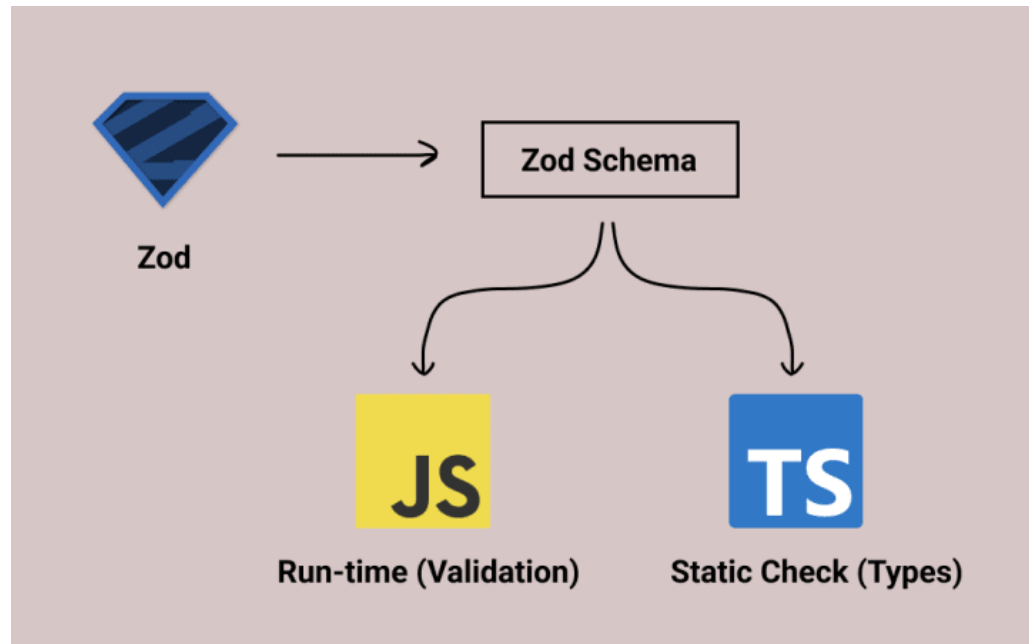


Traditional approach



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.

The screenshot shows a code editor on the left with the following code:

```
1 import { useState } from "react";
2 import { useForm } from "react-hook-form";
3 import Header from "../Header";
4
5 export function App() {
6   const { register, handleSubmit } = useForm();
7   const [data, setData] = useState("");
8
9   return (
10    <form onSubmit={handleSubmit((data) => setData)} >
11      <Header />
12      <input {...register("firstName")} placeholder="First name" />
13      <select {...register("category"), { required: true }} />
14        <option value="">Select...</option>
15        <option value="A">Option A</option>
16        <option value="B">Option B</option>
17      </select>
18      <textarea {...register("aboutYou")} placeholder="About you" />
19      <input type="submit" value="SUBMIT" />
20    </form>
21  );
22 }
23
```

On the right, the rendered form is shown with a "Render Count: 1" indicator. The form includes a header, a text input for "First name", a dropdown menu for "category", a text area for "About you", and a pink "SUBMIT" button.

The screenshot shows a form with a "Watching." label and a list of input fields. The first field is checked and labeled "Watching.". The second field is unchecked. The third field is checked and labeled "Watching.". The fourth field is unchecked.

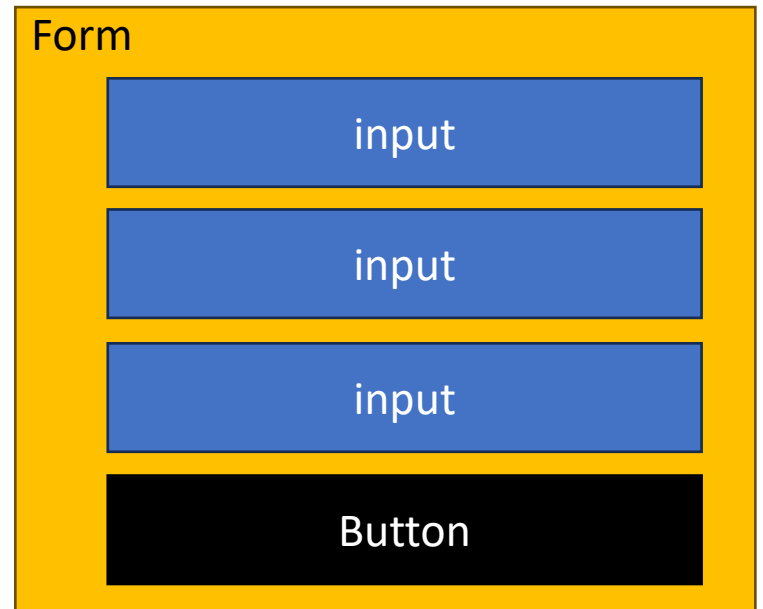
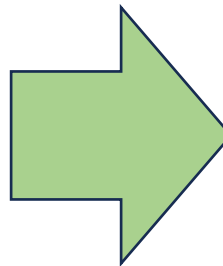
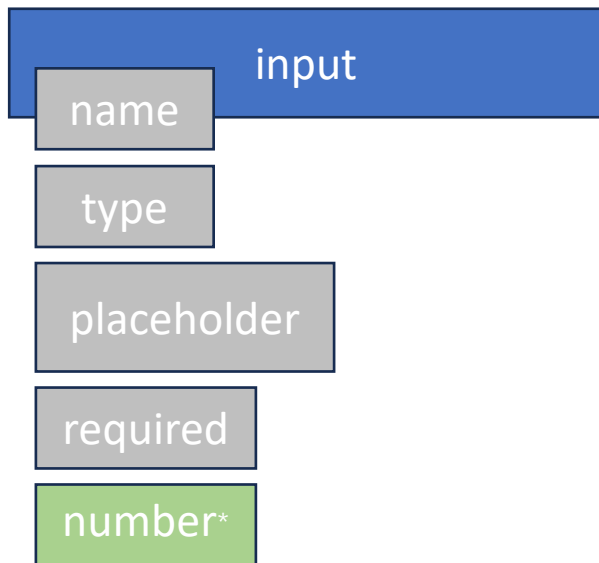


Example

> 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 > ...

```
1  import { FieldError, UseFormRegister } from "react-hook-form";
2
3  export type ValidFieldNames =
4    | "email"
5    | "githubUrl"
6    | "yearsOfExperience"
7    | "password"
8    | "confirmPassword";
9
10 export type FormData = {
11   email: string;
12   githubUrl: string;
13   yearsOfExperience: number;
14   password: string;
15   confirmPassword: string;
16 };
17
18 export type FormFieldProps = {
19   register: UseFormRegister<FormData>;
20   name: ValidFieldNames;
21   error: FieldError | undefined;
22   type: string;
23   placeholder: string;
24   valueAsNumber?: boolean;
25   required?: boolean;
26 };
27
```

```
1 import { FormFieldProps } from "./types";
2
3 const FormField: React.FC<FormFieldProps> = ({
4   type,
5   placeholder,
6   name,
7   register,
8   error,
9   valueAsNumber,
10  required,
11 }) => (
12  <>
13    <input
14      type={type}
15      placeholder={placeholder}
16      required={required}
17      {...register(name, { valueAsNumber })}
18    />
19    {error && <span>{error.message}</span>}
20  </>
21 );
22
23 FormField.defaultProps = {
24   required: false,
25 };
26
27 export default FormField;
28
```

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 > ...

```
1 import { useForm } from "react-hook-form";
2 import { FormData } from "./types";
3 import FormField from "./FormField";
4
5 function Form() {
6   const {
7     register,
8     handleSubmit,
9     formState: { errors },
10    setError,
11  } = useForm<FormData>();
12
13  const onSubmit = async (data: FormData) => {
14    console.log("SUCCESS", data);
15  };
16
17  return (
18    <form onSubmit={handleSubmit(onSubmit)}>
19      <div>
20        <h1>Zod & React-Hook-Form</h1>
21        <FormField
22          type="email"
23          placeholder="Email"
24          name="email"
25          register={register}
26          required
27          error={errors.email}
28        />
29
```

```
62      <button type="submit" className="submit-button">
63        Submit
64      </button>
65    </div>
66  </form>
67  );
68 }
69
70 export default Form;
71
```

30 <FormField

```
31   type="text"
32   placeholder="GitHub URL"
33   name="githubUrl"
34   register={register}
35   error={errors.githubUrl}
36 />
```

37 <FormField

```
38   type="number"
39   placeholder="Years of Experience (1 - 10)"
40   name="yearsOfExperience"
41   register={register}
42   error={errors.yearsOfExperience}
43   valueAsNumber
44 />
```

45 <FormField

```
46   type="password"
47   placeholder="Password"
48   name="password"
49   register={register}
50   error={errors.password}
51 />
```

52 <FormField

```
53   type="password"
54   placeholder="Confirm Password"
55   name="confirmPassword"
56   register={register}
57   error={errors.confirmPassword}
58 />
```

```
59 <button type="submit" className="submit-button">
60   Submit
61 </button>
62 </div>
63 </form>
64 );
65 }
```

http://127.0.0.1:3000/login

```
final > app > login > page.tsx > ...
 1  "use client";
 2  import Form from "@components/form/Form";
 3
 4  function Login() {
 5    return (
 6      <>
 7        <Form />
 8      </>
 9    );
10  }
11
12  export default Login;
13
```

Define a Form Schema with Zod

final > components > form > types.tsx > [x] FormData

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

Add new import

```
29 export const LoginSchema: ZodType<FormData> = z
30   .object({
31     email: z.string().email(),
32     githubUrl: z
33       .string()
34       .url()
35       .includes("github.com", { message: "Invalid GitHub URL" }),
36     yearsOfExperience: z
37       .number({
38         required_error: "required field",
39         invalid_type_error: "Years of Experience is required",
40       })
41       .min(1)
42       .max(10),
43     password: z
44       .string()
45       .min(8, { message: "Password is too short" })
46       .max(20, { message: "Password is too long" }),
47     confirmPassword: z.string(),
48   })
49   .refine((data) => data.password === data.confirmPassword, {
50     message: "Passwords do not match",
51     path: ["confirmPassword"], // path of error
52   });
53
```

Integrate Zod with React-Hook-Form for validation

Add new import

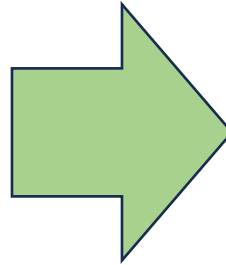
```
final > components > form > Form.tsx > Form
1  import { useForm } from "react-hook-form";
2  import { FormData, LoginSchema } from "./types";
3  import FormField from "./FormField";
4  import { zodResolver } from "@hookform/resolvers/zod";
5
6  function Form() {
7    const {
8      register,
9      handleSubmit,
10     formState: { errors },
11     setError,
12   } = useForm<FormData>({
13     resolver: zodResolver(LoginSchema),
14   });
15
```

Add new syntax

http://127.0.0.1:3000/login

Zod & React-Hook-Form

<input type="text" value="bfxv@d.c"/>	Invalid email
<input type="text" value="http://github.co"/>	Invalid GitHub URL
<input type="text" value="Years of Experience (1 - 1)"/>	Years of Experience is required
<input type="password" value="....."/>	Password is too short
<input type="password" value="....."/>	Passwords do not match
<input type="button" value="Submit"/>	



Zod & React-Hook-Form

<input type="text" value="bfxv@d.cc"/>	
<input type="text" value="http://github.com/"/>	
<input type="text" value="2"/>	
<input type="password" value="....."/>	
<input type="password" value="....."/>	
<input type="button" value="Submit"/>	

Console

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

>

Validate data from the **Register form.**



Let's Try

Guideline

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

Q & A