Vamos detalhar o funcionamento do componente fornecido:

Importações

1. **useFormValue**: Importado da URL fornecida, este hook permite armazenar e recuperar valores de formulário associados a um nome específico.
2. **addPropertyControls, ControlType**: Importados de "framer", são usados para definir os controles de propriedades que aparecerão no painel do Framer.

HOC (Higher-Order Component) - withFormInput

Este HOC envolve componentes de input para conectar seu valor ao estado gerenciado por `useFormValue`.

javascript
Copy code

```
export function withFormInput(Component) {
    return (props) => {
        const [value, setValue] = useFormValue(props.name, "")

        const handleChange = (event) => {
            setValue(event.target.value)
        }

        return <Component {...props} value={value}
onChange={handleChange} />
    }
}
```

- **useFormValue**: Recupera e define o valor de um campo de formulário associado a `props.name`.
- **handleChange**: Atualiza o valor quando o evento de mudança (`onChange`) é disparado.
- **Component**: Renderiza o componente de input passado, propagando as propriedades e adicionando `value` e `onChange`.

Componente de Slider

Este é o componente principal que renderiza um controle de deslize (slider) e mostra o valor atual.

javascript
Copy code

```
export function Slider(props) {
```

```
    const [value, setValue] = useFormValue(props.name,
props.defaultValue)

    const handleChange = (event) => {
        setValue(event.target.value)
    }

    return (
        <div style={{ display: "flex", alignItems: "center" }}>
            <input
                type="range"
                min={props.min}
                max={props.max}
                value={value}
                onChange={handleChange}
                name={props.name}
                style={{ marginRight: "10px" }}
            />
            <div
                style={{
                    width: "40px",
                    height: "30px",
                    display: "flex",
                    justifyContent: "center",
                    alignItems: "center",
                    backgroundColor: "#333",
                    color: "#fff",
                    borderRadius: "4px",
                }}
            >
                {value}
            </div>
        </div>
    )
}
```

- **useFormValue**: Inicializa o valor do slider com `props.defaultValue`.
- **handleChange**: Atualiza o valor do slider quando ocorre uma mudança.
- **input**: O controle deslizante que ajusta seu valor entre `props.min` e `props.max`.
- **div**: Exibe o valor atual do slider.

Controles de Propriedades para o Slider

Define as propriedades configuráveis no painel do Framer.

javascript
Copy code

```
addPropertyControls(Slider, {
    name: {
        type: ControlType.String,
        defaultValue: "sliderValue",
        title: "Name",
    },
    min: {
        type: ControlType.Number,
        defaultValue: 0,
        title: "Min Value",
    },
    max: {
        type: ControlType.Number,
        defaultValue: 100,
        title: "Max Value",
    },
    defaultValue: {
        type: ControlType.Number,
        defaultValue: 50,
        title: "Default Value",
    },
})
```

Exportação do Componente com HOC Aplicado

Envolve o `Slider` com `withFormInput` para criar um componente que gerencia seu valor de forma controlada.

javascript
Copy code

```
export const FormSlider = withFormInput(Slider)
```

Documentação

Comentário indicando o autor do código.

javascript

Copy code

```
/**
 * Code by @omatusaelhorch
 */
```

Instruções para Adicionar o Componente no Framer

1. Vá até **Assets**.
2. Selecione **Novo code component**.
3. Nomeie o componente e clique em **Create**.
4. Cole o código fornecido.
5. Arraste o componente criado dos assets para o seu projeto.

Se precisar de mais detalhes ou tiver alguma dúvida específica, envie um print do erro ou uma descrição mais detalhada da sua dúvida. Estou aqui para ajudar! Se precisar de mais suporte, pode entrar no nosso servidor do Discord: https://discord.gg/xEDVjePabx

Bases de referências que funciona:

```
import React from "react"
import { useFormValue } from "https://framer.com/m/Store-nhJr.js"
import { addPropertyControls, ControlType } from "framer"

// HOC para envolver os componentes de input e atualizar o valor
export function withFormInput(Component) {
    return (props) => {
        const [value, setValue] = useFormValue(props.name, props.defaultValue)

        const handleChange = (event) => {
            setValue(event.target.value)
        }

        return <Component {...props} value={value} onChange={handleChange} />
    }
}

// Componente de DatePicker
export function DatePicker(props) {
    const {
        value,
        onChange,
        name,
        color,
        borderColor,
        backgroundColor,
        textColor,
        width,
```

```
    height,
    borderRadius,
    icon,
  } = props

  return (
    <div
      style={{
        position: "relative",
        display: "inline-block",
        width: "100%",
      }}
    >
      <input
        type="date"
        value={value}
        onChange={onChange}
        name={name}
        style={{
          width: width,
          height: height,
          padding: "10px",
          borderRadius: borderRadius,
          border: `1px solid ${borderColor}`,
          backgroundColor: backgroundColor,
          color: textColor,
          fontSize: "16px",
          boxSizing: "border-box",
        }}
      />
      {icon && (
        <img
          src={icon}
          alt="icon"
          style={{
            position: "absolute",
            right: "10px",
            top: "50%",
            transform: "translateY(-50%)",
            width: "20px",
            height: "20px",
            pointerEvents: "none",
          }}
        />
      )}
    </div>
  )
}
```

```
// Property controls para o DatePicker
addPropertyControls(DatePicker, {
    name: {
        type: ControlType.String,
        defaultValue: "datePickerValue",
        title: "Name",
    },
    defaultValue: {
        type: ControlType.String,
        defaultValue: new Date().toISOString().substr(0, 10),
        title: "Default Value",
    },
    color: {
        type: ControlType.Color,
        title: "Text Color",
        defaultValue: "#000000",
    },
    borderColor: {
        type: ControlType.Color,
        title: "Border Color",
        defaultValue: "#007BFF",
    },
    backgroundColor: {
        type: ControlType.Color,
        title: "Background Color",
        defaultValue: "#FFFFFF",
    },
    textColor: {
        type: ControlType.Color,
        title: "Text Color",
        defaultValue: "#000000",
    },
    width: {
        type: ControlType.String,
        title: "Width",
        defaultValue: "100%",
    },
    height: {
        type: ControlType.String,
        title: "Height",
        defaultValue: "40px",
    },
    borderRadius: {
        type: ControlType.Number,
        title: "Border Radius",
        defaultValue: 4,
    },
```

```
    icon: {
        type: ControlType.Image,
        title: "Icon",
    },
    codeBy: {
        type: ControlType.String,
        title: " ",
        defaultValue: "Code by UncodeStack and @omatusaelhorch",
        readOnly: true,
        description:
            "Este componente foi criado por UncodeStack e @omatusaelhorch.",
    },
})

// Exportando o DatePicker com HOC aplicado
export const FormDatePicker = withFormInput(DatePicker)

// Documentação
/**
 * Code by @omatusaelhorch
 */
```