**Forms with traditional react approach  
Forms using 3rd Party  
Formik  
  initialValues  
  onSubmit  
  handleChange  
  handleBlur  
  
                            Formik Validations  
- You configure validation for every element in page by using validation function.  
- Formik uses a validation function  
  
Syntax:  
            function ValidationFunction(formDetails)  
            {  
                 const errors = { };  
                 ..write logic for verifying every field...  
                 return errors;  
            }   
  
            useFormik({  
                initialValues: { },  
                validate: ValidationFunction  
            })  
  
- You can access error message by using  
  
            { formik.errors.Name }  
  
  
Ex: Simple Validation  
 *import { useFormik } from "formik";  
  
export default function FormikValidation()  
{  
    function VerifyUserDetails(userDetails)  
    {  
        const errors = {};  
        if(userDetails.UserName==""){  
            errors.UserName = "User Name Required";  
        }  
        if(userDetails.Age=="") {  
            errors.Age = "Age Required";  
        }  
        if(userDetails.Email=="") {  
            errors.Email = "Email Required";  
        }  
        return errors;  
    }  
    const formik = useFormik({  
        initialValues: {  
            UserName: '',  
            Age: 0,  
            Email:''  
        },  
        validate: VerifyUserDetails,  
        onSubmit: values => {  
            alert(JSON.stringify(values));  
        }  
    })  
    return(  
        <div className="container-fluid">  
            <form onSubmit={formik.handleSubmit}>  
                <h2>Register User</h2>  
                <dl>  
                    <dt>User Name</dt>  
                    <dd><input name="UserName" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.UserName}</dd>  
                    <dt>Age</dt>  
                    <dd><input name="Age" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Age}</dd>  
                    <dt>Email</dt>  
                    <dd><input name="Email" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Email}</dd>  
                </dl>  
                <button>Register</button>  
            </form>  
        </div>  
    )  
}*Ex: Multiple Validations for every field  
  
*import { useFormik } from "formik";  
  
export default function FormikValidation()  
{  
    function VerifyUserDetails(userDetails)  
    {  
        const errors = {};  
        if(userDetails.UserName==""){  
            errors.UserName = "User Name Required";  
        } else if(userDetails.UserName.length<4){  
            errors.UserName = "Name too short..";  
        } else if(userDetails.UserName.length>10){  
            errors.UserName = "Name too long..";  
        }  
  
        if(userDetails.Age=="") {  
            errors.Age = "Age Required";  
        } else if(isNaN(userDetails.Age)){  
            errors.Age = "Age must be a Number";  
        }  
  
        if(userDetails.Email=="") {  
            errors.Email = "Email Required";  
        }else if(userDetails.Email.indexOf("@")<=2) {  
            errors.Email = "Invalid Email";  
        }  
  
        if(userDetails.Mobile=="") {  
            errors.Mobile = "Mobile Required";  
        }  
        else if(userDetails.Mobile.match(/\+91\d{10}/)) {  
            errors.Mobile = "";  
        } else {  
            errors.Mobile = "Invalid Mobile"  
        }  
  
        return errors;  
    }  
    const formik = useFormik({  
        initialValues: {  
            UserName: '',  
            Age: 0,  
            Email:'',  
            Mobile:''  
        },  
        validate: VerifyUserDetails,  
        onSubmit: values => {  
            alert(JSON.stringify(values));  
        }  
    })  
    return(  
        <div className="container-fluid">  
            <form onSubmit={formik.handleSubmit}>  
                <h2>Register User</h2>  
                <dl>  
                    <dt>User Name</dt>  
                    <dd><input name="UserName" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.UserName}</dd>  
                    <dt>Age</dt>  
                    <dd><input name="Age" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Age}</dd>  
                    <dt>Email</dt>  
                    <dd><input name="Email" onChange={formik.handleChange} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Email}</dd>  
                    <dt>Mobile</dt>  
                    <dd><input type="text" name="Mobile" onChange={formik.handleChange} /></dd>  
                    <dd className="text-danger">{formik.errors.Mobile}</dd>  
                </dl>  
                <button>Register</button>  
            </form>  
        </div>  
    )  
}*  
                            Yup Library**

**- It simplifies the form and its validation.  
- Yup library provides validation schemas.  
- Every validation schema [structure] defines the type and range of value.  
- Validation Schema is Data Structure.  
- Explicitly you don't need any function to verify the data type or range.  
                         
                    JavaScript  
                    - isNaN  
                    - typeof  
                    - instanceof  
                    - in : It defined the property is available or not.   
  
                var x = "true";  
                var result = (x=="true")true?false;  
  
- Yup Library provides schemas to verify type  
        BooleanSchema            boolean()  
        NumberSchema            number()  
        StringSchema etc..        string()  
  
- Yup library provides functions for verifying range and pattern.  
        min()  
        max()  
        maxlength()  
        minlength()  
        required()  
        pattern()  
        email() etc..  
  
1. Install Yup library for project  
  
        >npm install yup --save  
  
2. Import Validation Schemas in component  
  
        import  { BooleanSchema, NumberSchema,.. } from 'yup';  
  
        import  \* as yup from 'yup'; //importing all schemas.  
  
3. Formik can use Yup validation schema by using  
        "validationSchema" property  
  
     const formik = useFormik({  
            initialValues: { },  
          validationSchema: yup.object({  
                 FieldName: yup.DataType().required().min(4).max(10)  
            })  
    })  
  
4. You have to bind the fields with HTML elements  
  
    <input type="text" {...formik.getFieldProps("Name")}>  
    <dd>{formik.errors.Name}</dd>**

**Note: … means rest parameter, it means it(a parameter) can take multiple arguments.  
  
Ex: Yup Validation  
  
*import { useFormik } from "formik";  
import  \* as yup from 'yup';  
  
export default function YupValidation()  
{  
    const formik = useFormik({  
        initialValues: {  
            UserName: '',  
            Email: '',  
            Age: 0  
        },  
        validationSchema: yup.object({  
            UserName: yup.string()  
                         .required('User Name Required')  
                         .min(4, 'Name too short')  
                         .max(10, 'Name too Long..'),  
            Email: yup.string()  
                      .required('Email Required')  
                      .email('Invalid Email'),  
            Age: yup.number()  
                    .required('Age Required')  
        }),  
        onSubmit : values => {  
            alert(JSON.stringify(values));  
        }  
    })  
    return(  
        <div className="container-fluid">  
            <form onSubmit={formik.handleSubmit}>  
                <h2>Register User</h2>  
                <dl>  
                    <dt>User Name</dt>  
                    <dd><input {...formik.getFieldProps("UserName")} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.UserName}</dd>  
                    <dt>Email</dt>  
                    <dd><input {...formik.getFieldProps("Email")} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Email}</dd>  
                    <dt>Age</dt>  
                    <dd><input {...formik.getFieldProps("Age")} type="text"/></dd>  
                    <dd className="text-danger">{formik.errors.Age}</dd>  
                </dl>  
                <button>Register</button>  
            </form>  
        </div>  
    )  
}***