Form Validation and Events

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**1**	**Week No:**	**11**	**Semester:**	
Lecturer:	*Sazzad Hossain; sazzad@aiub.edu*				

Lecture Outline

- ✓ Form Validation in Next.js
- ✓ HTML form validation
- ✓ Vanilla JS Form Validation
- ✓ External Libraries
- ✓ Logical AND (&&)
- ✓ Redirecting using useRouter() Hook

Form Validation in Next.js

- Form validation is the process of ensuring that user inputs are accurate and complete
- Confirm to the required format before submitting the form.
- It helps prevent invalid data from being submitted to the server.
- Next.js supports built-in HTML form validation attributes like "required," "type," and "pattern“
- Vanilla JavaScript can be used to implement custom validation logic.

HTML form validation

```
import React from 'react';

export default function IndexPage  ()  {
  const handleSubmit = (e) => {
    e.preventDefault();
    console.log('Form submitted successfully!');
  };

  return (
    <>
      <h1>Form Validation</h1>
      <form onSubmit={handleSubmit}>
        <div>
          <label htmlFor="name">Name:</label>
          <input type="text" id="name" required />
        </div>
        <div>
          <label htmlFor="email">Email:</label>
          <input type="email" id="email" required />
        </div>
        <div>
          <label htmlFor="password">Password:</label>
          <input type="password" id="password" minLength="8" required />
        </div>
        <div>
          <button type="submit">Submit</button>
        </div>
      </form>
    </>
  );
};
```

required: Ensures the field must be filled out before submitting the form.
type="email": Validates that the email input follows the correct email format.
minLength="8": Specifies that the password must be at least 8 characters long.

HTML form validation

- Full guide of HTML built in Validation can be found here;

https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation

Vanilla JS Form Validation

```
import { useState } from 'react';
import Layout from "./Layout/layout";
import Title from "./Layout/title";

export default function LoginForm () {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleChangeEmail = (e) => {
    setEmail(e.target.value);
  };

  const handleChangePassword = (e) => {
    setPassword(e.target.value);
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    // Perform form validation
    if (!email || !password) {
      setError('Email and password are required');
    } else if (!isValidEmail(email)) {
      setError('Invalid email address');
    } else {
      console.log({ email, password });
      setEmail('');
      setPassword('');
      setError('');
    }
  };
```

```
  const isValidEmail = (email) => {
    const emailPattern = /^\S+@\S+\.\S+$/;
    return emailPattern.test(email);
  };

  return (
    <>
     <Title page="Login"> </Title>
<Layout>
      <h1>Login</h1>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Email</label>
          <input
            type="email"
            name="email"
            value={email}
            onChange={handleChangeEmail}
          />
        </div>
        <div>
          <label>Password</label>
          <input
            type="password"
            name="password"
            value={password}
            onChange={handleChangePassword}
          />
        </div>
        {error && <p>{error}</p>}
        <button type="submit">Login</button>
      </form>
      </Layout>
    </>
  );};
```

Logical AND (&&)

The **logical AND (&&)** (logical conjunction) operator for a set of boolean operands will be **true if and only if all the operands are true**. Otherwise it will be false.

```
const a = 3;
const b = -2;

console.log(a > 0 && b > 0);
// What is the output
```

External Libraries for Form Validation

Following are some external libraries for Form validation

- **React Hook Form**
- **Formik**
- **Yup**

Redirecting using useRouter() Hook

- **useRouter hook** from **next/router** to get access to the router object in functional component
- **push** method to navigate to another page on the event.

```
import { useRouter } from 'next/router';

function MyComponent() {
  const router = useRouter();

  const handleClick = () => {
    router.push('/new-page');
  };

  return (
    <button onClick={handleClick}>Go to new page</button>
  );
}
```

Redirecting using useRouter() Hook

- **router.push** method to pass query parameters to the new page

```
import { useRouter } from 'next/router';

function MyComponent() {
  const router = useRouter();

  const handleClick = () => {
  router.push({
  pathname: '/new-page',
  query: {
    name: 'Source',
    count: 30,
  },
  });
  };

  return (
    <button onClick={handleClick}>Go to new page</button>
  );
}
```

References

1. **W3Schools Online Web Tutorials, URL: http://www.w3schools.com**
2. **Next.js, URL:** https://nextjs.org/
3. **Mozilla Developer Networks, URL:** https://developer.mozilla.org/

Thank You!