

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

الحمد لله رب العالمين ، نحمده ونستعينه
ونستغفره والصلاة والسلام على أشرف
الأنبياء والمرسلين نبينا محمد ﷺ



پوهنتون کاردان
KARDAN UNIVERSITY

React Foundation

Lecture – 1

Khalilullah Akbari
k.akbari@kardan.edu.af

Foundations

- Introduction to React
- Why React
- JSX
- Virtual DOM



Introduction to React

- JavaScript library for building UI
- Component-based
 - A component is:
 - A **function (or class)**
 - That returns UI (JSX)
 - And can be reused anywhere
- Maintained by Meta



Why React

Single Page Applications (SPA)

- Client-side rendering (CSR)
- Uses JavaScript to update UI
- Communicates with server via APIs (usually JSON)
- Faster navigation after initial load

Reusable components

Efficient updates with Virtual DOM

- Creates a Virtual DOM tree
- When state/props change → creates a new Virtual DOM
- Compares it with the previous one (diffing)
- Updates only the changed parts in the real DOM



JSX

- JavaScript + HTML syntax
 - Must return single parent
 - Use {} for expressions
-
- JSX = HTML-like syntax in JavaScript
 - Used to describe UI
 - Must follow JSX rules



Demo

- Create simple component
- Render it
- Show rendering flow

```
npm create vite@latest my-app  
cd my-app  
npm install  
  
npm run dev
```



Part 2: Components in Depth

- Functional vs Class
- Props
- Validation
- Composition



Functional vs Class

- **Functional Components (Modern)**
- Simple JavaScript functions
- Use **Hooks** (e.g., useState, useEffect)
- Less code, easier to read
- Preferred in modern React

```
function Welcome() {  
  return <h1>Hello</h1>;  
}
```



Functional vs Class

- ES6 classes
- Use this and lifecycle methods
- More complex and verbose

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello</h1>;
  }
}
```

Feature	Functional	Class
Syntax	Simple	Complex
State	Hooks	this.state
Lifecycle	Hooks	Lifecycle methods
Usage Today	✓ Recommended	✗ Legacy



Props

- **What are Props?**
- Props = **inputs to components**
- Used to **pass data from parent** → **child**

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

```
// Usage  
<Welcome name="Ali" />
```



Props Validation

- **Why validate props?**
- Ensure components receive **correct data types**
- Prevent runtime errors
- Improve code reliability
-

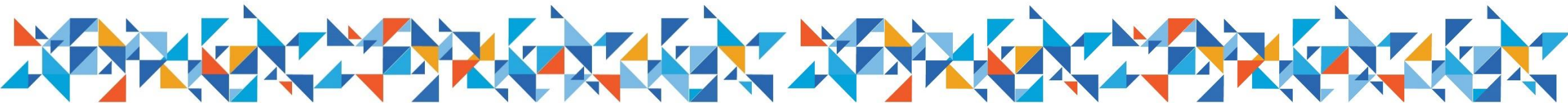
- **1. PropTypes (Runtime validation)**
- Checks props **during execution**
- Useful in JavaScript projects

npm install prop-types

```
import PropTypes from "prop-types";

function User({ name, age }) {
  return <p>{name} ({age})</p>;
}

User.propTypes = {
  name: PropTypes.string.isRequired,
  age: PropTypes.number
};
```



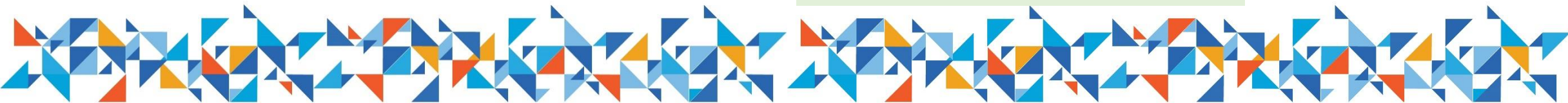
Composition

- **What is Component Composition?**
- Building UI by **combining smaller components**
- Instead of one large component → multiple reusable pieces

- **Key Idea**

- Components = **building blocks**
- Combine them to build complex UI

```
function Header() {  
  return <h1>My App</h1>;  
}  
  
function App() {  
  return (  
    <div>  
      <Header />  
      <p>Welcome to the app</p>  
    </div>  
  );  
}
```



Reusable Structure & Children Props

- **1. Reusable Structure**
 - Same component used in different places
 - Reduces duplication
- **2. Children Props**
 - Used to pass content inside components

```
function Card({ children }) {  
  return <div  
    className="card">{children}</div>;  
}
```

```
<Card>  
  <h2>Title</h2>  
  <p>Description</p>  
</Card>
```



Part 3: Interactivity

- Events
- Conditional Rendering
- Lists & Keys



Event Handling

- **What are events?**
- Events are user interactions like:
 - Click
 - Typing
 - Form changes
- **Common React Events**
- `onClick` → button clicks
- `onChange` → input changes

```
function App() {  
  return <button onClick={() => alert("Clicked!")}>Click  
me</button>;  
}
```



Conditional Rendering

- Showing different UI based on **conditions**
- Uses:
 - if statements
 - ternary operator (?:)

```
function Message({ isLoggedIn }) {  
  if (isLoggedIn) {  
    return <h1>Welcome back</h1>;  
  }  
  return <h1>Please log in</h1>;  
}
```

```
function Message({ isLoggedIn }) {  
  return (  
    <h1>  
      {isLoggedIn ? "Welcome back" : "Please log in"}  
    </h1>  
  );  
}
```



Lists & Keys

- Use JavaScript map() function
- Convert array → list of UI elements

```
function App() {  
  const students = ["Ali", "Sara", "John"];  
  
  return (  
    <ul>  
      {students.map((name) => (  
        <li>{name}</li>  
      ))}  
    </ul>  
  );  
}
```



useState

- A **Hook** used to add state to functional components
- Allows components to **store and update data**

- **Basic Syntax**

`const [state, setState] = useState(initialValue);`

- state → current value
- setState → function to update value
- initialValue → starting value

```
import { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>{count}</p>
      <button onClick={() => setCount(count + 1)}>
        Increase
      </button>
    </div>
  );
}
```





پوهنتون کاردان
KARDAN UNIVERSITY

Thank You