



Skills
Network

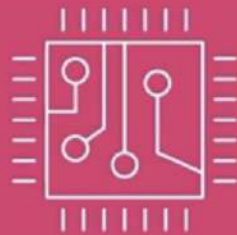
**Welcome to Working with
Function Components:
Props and Event Handling**

IBM

What you will learn



Explain the value of props and default props in components

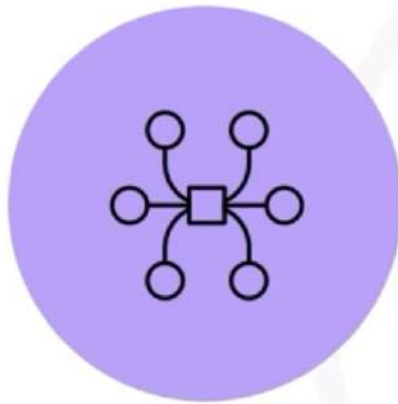


Summarize how to use props to pass data between components

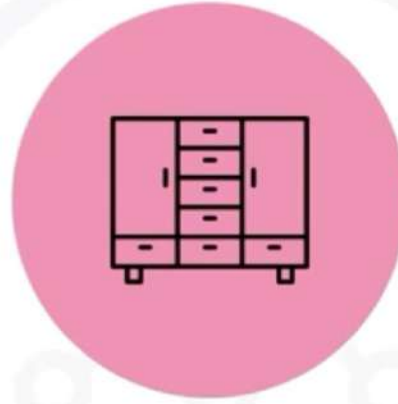


Discuss how events work in conjunction with props

Introduction to props



Data flow



Props store
data



Pass attributes

Read only data



Immutable



A child cannot
change attributes

Code example

```
function App() {  
  return (  
    <div>  
      <EmployeeData name="John" />  
    </div>  
  );  
}
```

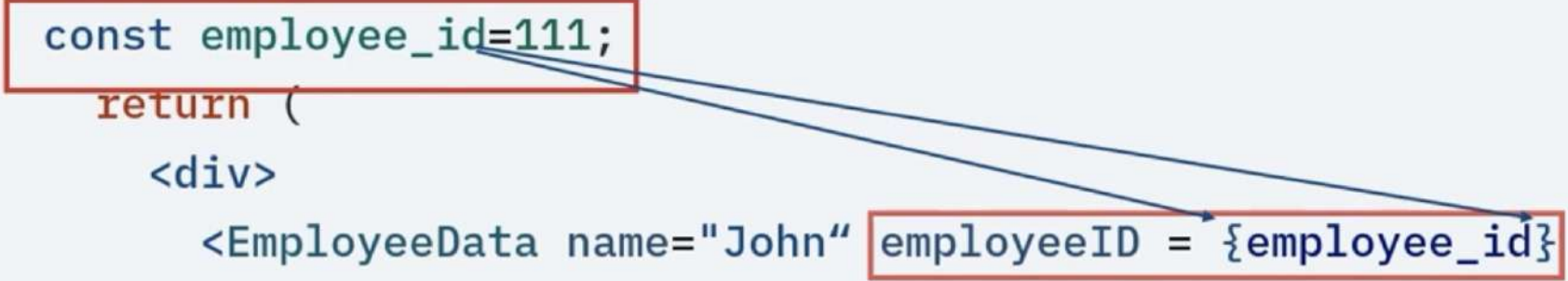
EmployeeData component

```
const EmployeeData =(props) =>{  
  return (  
    <>  
      <h1> {props.name} </h1>  
    </>  
  );  
}
```

name = "John"

Data formats

```
function App() {  
  const employee_id=111;  
  return (  
    <div>  
      <EmployeeData name="John" employeeID = {employee_id}  
      dept_id={567} />  
    </div>  
  );  
}
```



The diagram illustrates the flow of data from a JavaScript variable to a JSX element. A red box highlights the variable `const employee_id=111;` in the function `App()`. Two blue arrows originate from this box: one points to the `employeeID` prop in the `<EmployeeData>` component, and the other points to the `{employee_id}` expression within the `employeeID` prop's value. A second red box highlights the `employeeID = {employee_id}` part of the JSX element, showing how the variable's value is passed to the component.

Similarity to objects

Foundation for rendering UI elements

```
const EmployeeData =(props) =>{  
  return (  
    <>  
      <h1> {props.name} </h1>  
      <div> {props.employeeID } </div>  
      <div> {props.dept_id } </div>  
    );  
  }  
}
```

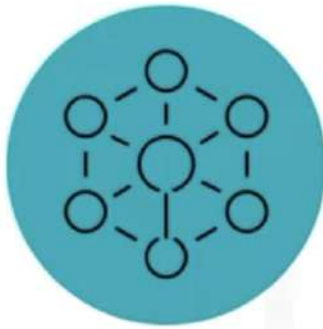

Default values

- .
- .
- .

```
EmployeeData.defaultProps = {  
  dept_name: "Human Resources",  
};
```

```
export default EmployeeData;
```

Default value purpose



Consistent rendering

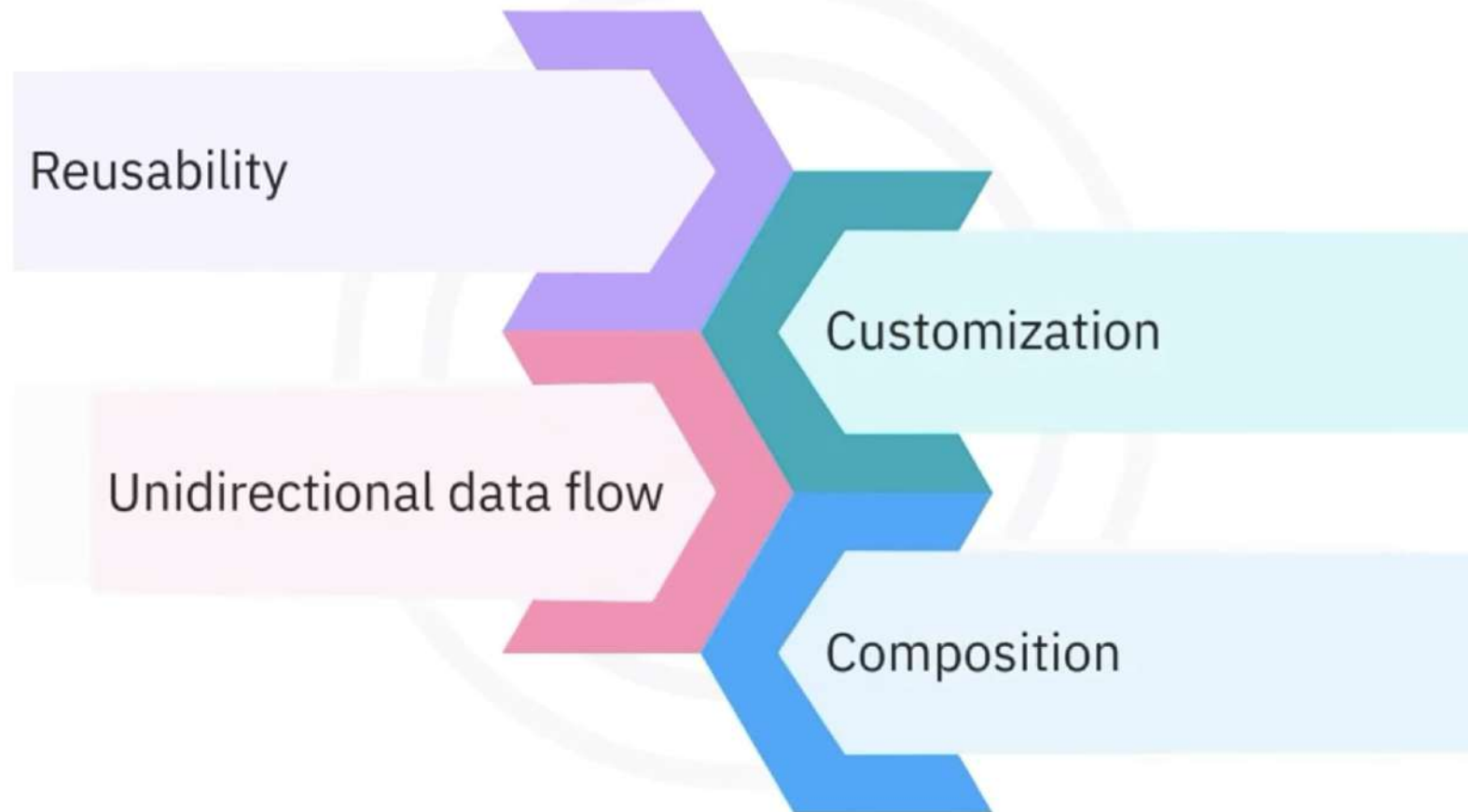


Prop data not
supplied in the parent

Default value syntax

```
.  
.   
.   
EmployeeData.defaultProps = {  
  dept_name: "Human Resources",  
};  
export default EmployeeData;
```

Principles



Principles explained

Reusability

- Minimize code duplication

Customization

- Simple yet robust

Unidirectional data flow

- Code readability
- Testing and debugging

Composition

- Keeps code simple
- Build complex UIs

Props and events

```
function App() {  
  const employee_id=111;  
  return (  
    <div>  
      <EmployeeData name="John" employeeId =  
        {employee_id} dept_id={567} increase={10000}/>  
    </div>  
  );  
}
```

Click event example

```
const Props = (props) => {  
  const [showIncrease, setShowIncrease] = useState(false);  
  return (  
    <>  
      <h1>{props.name}</h1>  
      <h2>{props.employeeId}</h2>  
      <h2>{props.dept_is}</h2>  
      <div>{props.dept_name}</div>  
      <button onClick={() => setShowIncrease(!showIncrease)}>  
        {showIncrease ? 'Hide' : 'Display'} the annual increase  
      </button>  
      {showIncrease && <div>{props.increment}</div>}  
    </>  
  );  
};
```

Output

showIncrease = false	showIncrease = true
<p>John</p> <p>111</p> <p>Human Resources</p> <p>Display annual increase</p>	<p>John</p> <p>111</p> <p>Human Resources</p> <p>Hide annual increase</p> <p>\$10,000</p>

Recap

In this video, you learned that:

- A prop stores the attributes of a component and allows you to pass these attributes from parent to child
- You pass a props object as a parameter into a component. Define the data using the dot operator in the format ``props.attributeName``
- Props principles include reusability, unidirectional data flow, customization, and composition
- You can use the ``useState`` hook in conjunction with an event to control the UI in a child component