

TI2104 – Pengembangan Aplikasi Web

State dan Props:

- Pengertian dan perbedaan antara state dan props
- Pengelolaan state & props dalam komponen
- Event, Set State
- Form Handling



Strata - 1

Teknologi Informasi



Pengenalan State

Chapter 1



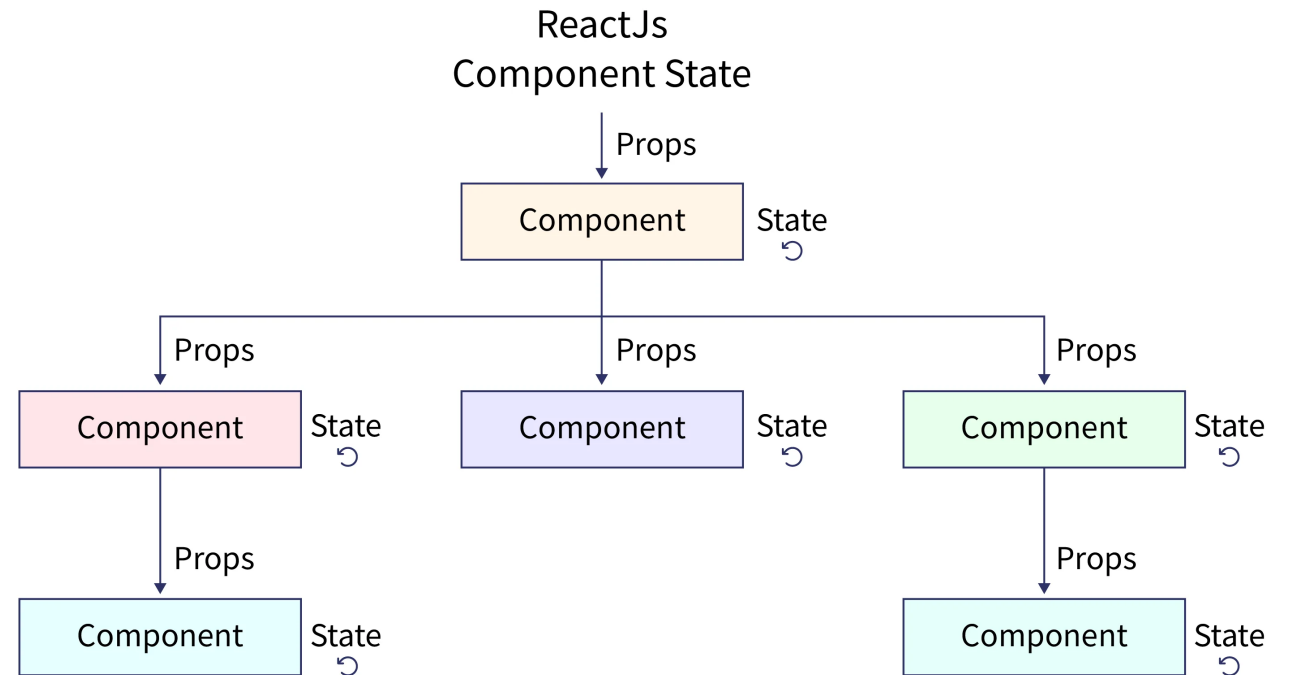
State dalam ReactJs

- State adalah suatu **objek yang menyimpan informasi** tentang komponen yang dapat **berubah** seiring waktu.
- State memungkinkan komponen untuk merespons **perubahan dan merender ulang UI** berdasarkan perubahan tersebut.



Mengapa Menggunakan State?

1. Untuk menyimpan **informasi yang berubah-ubah dalam komponen.**
2. Untuk **merender ulang komponen** ketika ada perubahan data.
3. Untuk membuat aplikasi yang **dinamis dan interaktif.**




SCALER
Topics

Membuat State

(Pada komponen berbasis kelas)

Dalam komponen berbasis kelas, **state adalah objek yang menyimpan nilai-nilai yang berkaitan dengan komponen** dan dapat berubah seiring waktu. Perubahan pada state dapat memicu ulang render komponen.

Inisialisasi State



```
1  class MyComponent extends React.Component {
2      constructor(props) {
3          super(props);
4          this.state = {
5              count: 0
6          };
7      }
8  }
```

State dalam Class Component (lanjutan)

Mengakses **state** dengan menggunakan **this.state**

Mengakses State



```
1  render() {  
2    return <h1>{this.state.count}</h1>;  
3  }
```

UNIVERSITAS
MIKROSKIL

State dalam Class Component (lanjutan)

Untuk memperbarui state, harus menggunakan **this.setState()**.

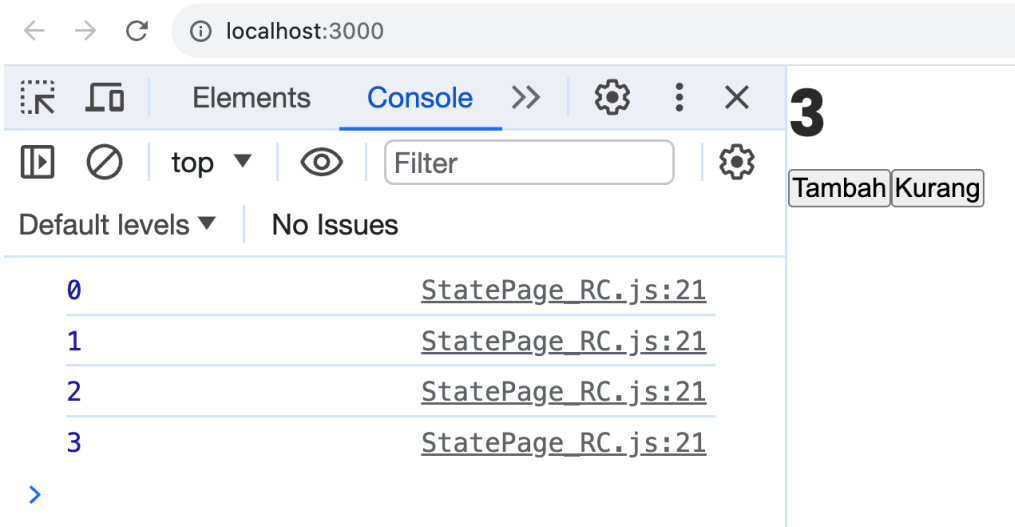
Ini memberi tahu React bahwa **state telah berubah, dan komponen harus di-render ulang.**

Memperbarui State



```
1  handleIncrement = () => {  
2    this.setState({ count: this.state.count + 1 });  
3  };  
4  
5  handleDecrement = () => {  
6    this.setState({ count: this.state.count - 1 });  
7  };
```

Contoh State dalam Class Component



```
1 import React from "react";
2
3 class StatePageRC extends React.Component {
4   constructor(props) {
5     super(props);
6     this.state = {
7       count: 0,
8     };
9   }
10
11   handleIncrement = () => {
12     this.setState({ count: this.state.count + 1 });
13   };
14
15   handleDecrement = () => {
16     this.setState({ count: this.state.count - 1 });
17   };
18
19   render() {
20     console.log(this.state.count);
21     return (
22       <div>
23         <h1>{this.state.count}</h1>
24         <button onClick={this.handleIncrement}>Tambah</button>
25         <button onClick={this.handleDecrement}>Kurang</button>
26       </div>
27     );
28   }
29 }
30
31 export default StatePageRC;
```


State dengan Hooks

(Fungsional Komponen)

Hooks adalah fitur yang diperkenalkan **dalam React 16.8** yang **memungkinkan menggunakan state dan fitur React lainnya tanpa menulis kelas**.

useState Hook

useState adalah Hook yang memungkinkan menambahkan state React ke komponen fungsi.

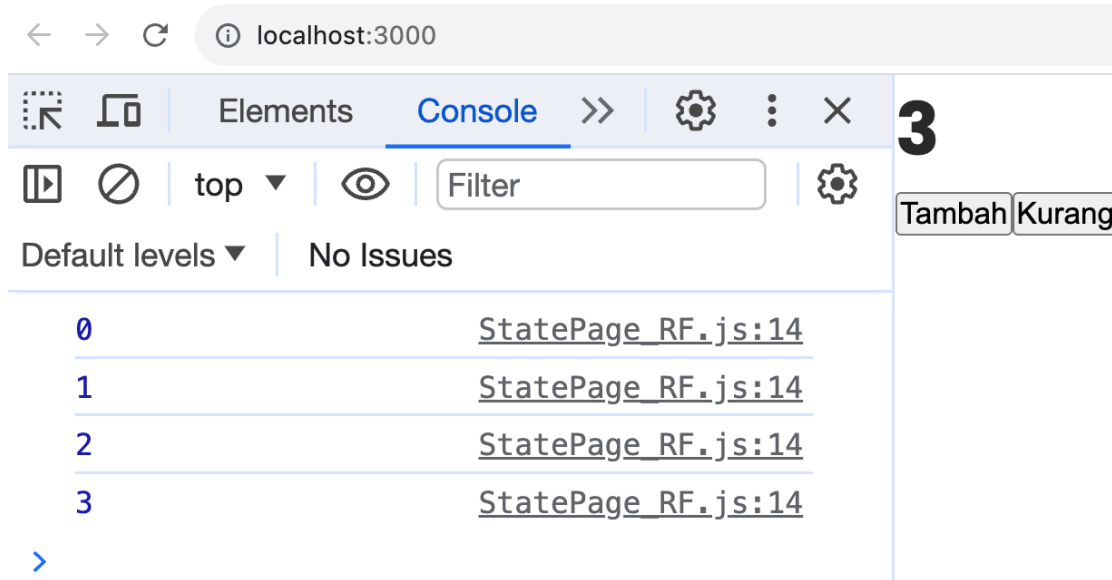
Sintaks useState



```
1 //state dengan Hook
2 const [state, setState] = useState(initialState);
3
4 // contoh:
5 const [count, setCount] = useState(0);
```

- **state** adalah variabel yang menyimpan nilai state saat ini.
- **setState** adalah fungsi yang digunakan untuk memperbarui state.
- **initialState** adalah nilai awal dari state.

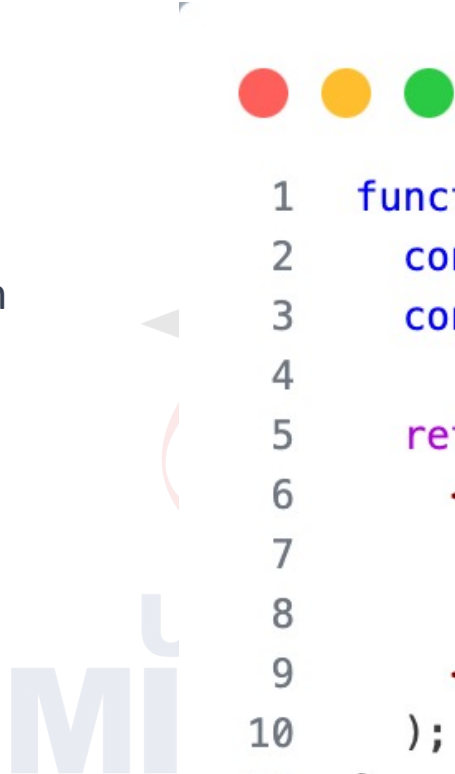
State dengan Hooks | Contoh (Fungsional Komponen)



```
1  import React, { useState } from "react";
2
3  const StatePageRF = () => {
4    const [count, setCount] = useState(0);
5
6    const handleIncrement = () => {
7      setCount(count + 1);
8    };
9
10   const handleDecrement = () => {
11     setCount(count - 1);
12   };
13
14   return (
15     <div>
16       <h1>{count}</h1>
17       <button onClick={handleIncrement}>Tambah</button>
18       <button onClick={handleDecrement}>Kurang</button>
19     </div>
20   );
21 };
22
23 export default StatePageRF;
24
```

State dengan Hooks | Contoh (Fungsional Komponen)

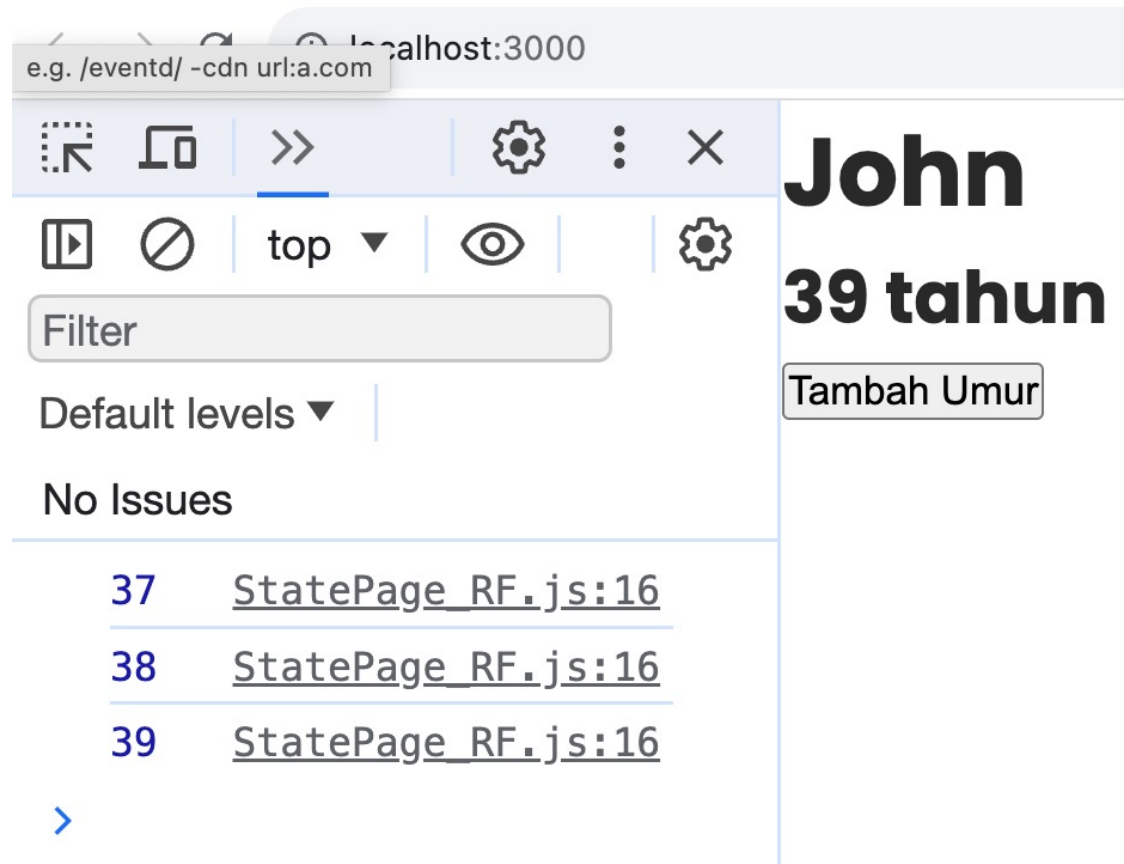
useState beberapa kali dalam satu komponen untuk mendeklarasikan lebih dari satu state.



```
1  function UserInfo() {
2    const [name, setName] = useState("John");
3    const [age, setAge] = useState(30);
4
5    return (
6      <div>
7        <h1>{name}</h1>
8        <h2>{age} tahun</h2>
9      </div>
10   );
11 }
12
13 export default UserInfo;
```

State dengan Hooks | Contoh (Fungsional Komponen)

State dengan Objek



```
1 import React, { useState } from "react";
2
3 function Profile() {
4   const [profile, setProfile] = useState({
5     name: "John",
6     age: 30,
7   });
8
9   const handleAgeIncrement = () => {
10     setProfile((prevProfile) => ({
11       ...prevProfile,
12       age: prevProfile.age + 1,
13     }));
14   }
15
16   return (
17     <div>
18       <h1>{profile.name}</h1>
19       <h2>{profile.age} tahun</h2>
20
21       <button onClick={handleAgeIncrement}> Tambah Umur </button>
22     </div>
23   );
24 }
25
26 export default Profile;
27
```

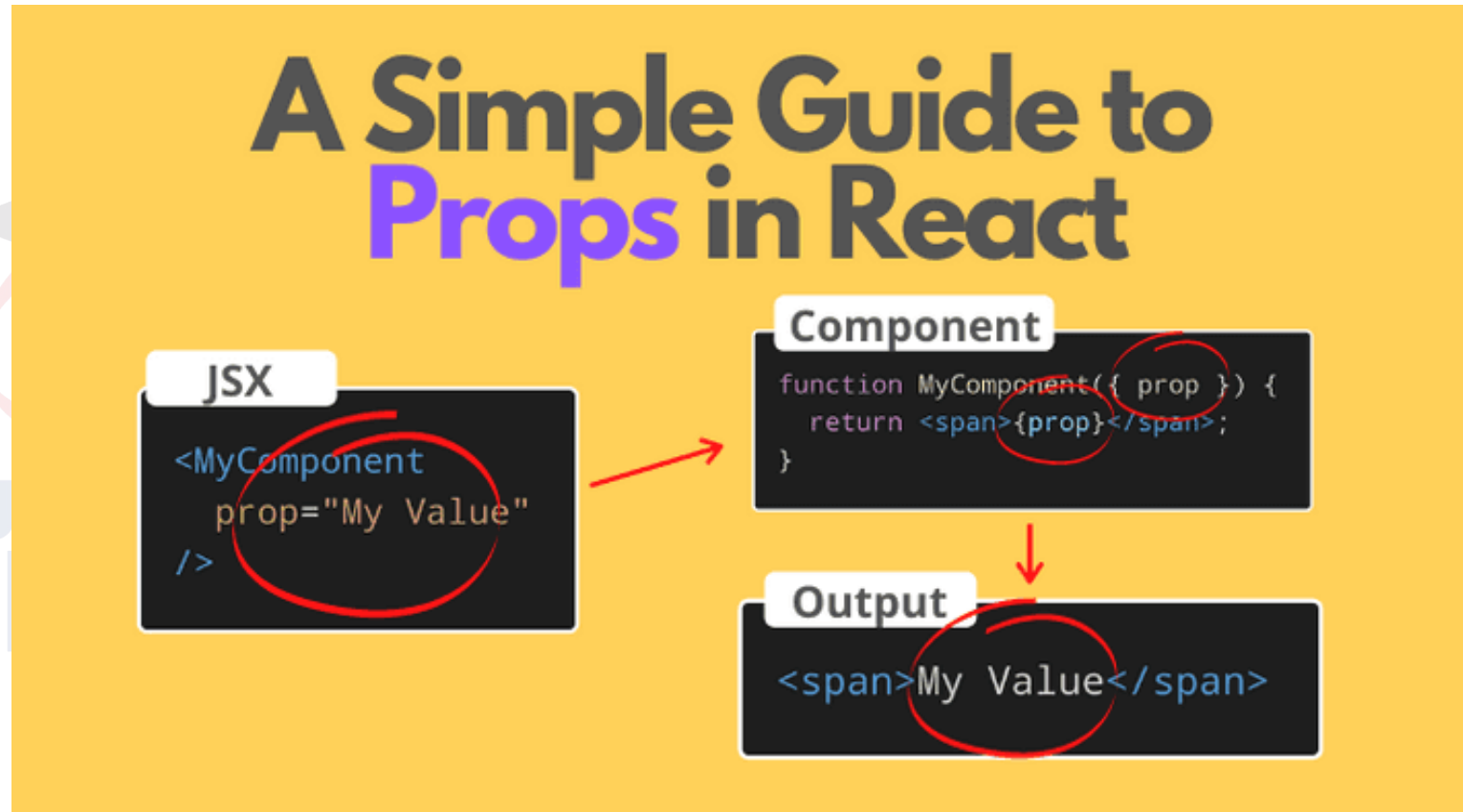


Pengenalan Props

Chapter 2

Props pada ReactJs

- **Props** (singkatan dari "properties") adalah mekanisme khusus dalam React yang **memungkinkan komponen untuk menerima data dari komponen induknya**.
- **Props bersifat read-only**, yang berarti sebuah komponen tidak dapat mengubah nilai props yang diterimanya.



Menggunakan Props

Props dapat diteruskan ke komponen dalam bentuk atribut, mirip dengan atribut HTML.

Dalam komponen, props dapat diakses melalui **objek props**.

Dalam contoh di atas, kita meneruskan **prop name** dengan nilai "John" ke komponen Welcome.

Di dalam komponen Welcome, kita mengakses prop tersebut dengan `props.name`.



```
1  function Welcome(props) {  
2    return <h1>Hello, {props.name}</h1>;  
3  }  
4  
5  function Props1() {  
6    return <Welcome name="John" />;  
7  }  
8  
9  export default Props1;  
10
```

Props untuk meneruskan fungsi

Dalam contoh di atas, kita meneruskan fungsi **handleClick** dari komponen App ke komponen Button melalui props.



```
1  function CustomButton(props) {
2    return <button onClick={props.handleClick}>Click Me</button>;
3  }
4
5  function Props2() {
6    const handleClick = () => {
7      alert("Button clicked!");
8    };
9
10   return <CustomButton handleClick={handleClick} />;
11 }
12
13 export default Props2;
```


Penggunaan State dan Props

Chapter 3

UNIVERSITAS
MIKROSKIL



State dan Props

Dalam aplikasi React yang kompleks, seringkali perlu **menggabungkan state dan props**.

Komponen mungkin menerima beberapa data melalui props dan kemudian menggunakannya bersama dengan state lokal.

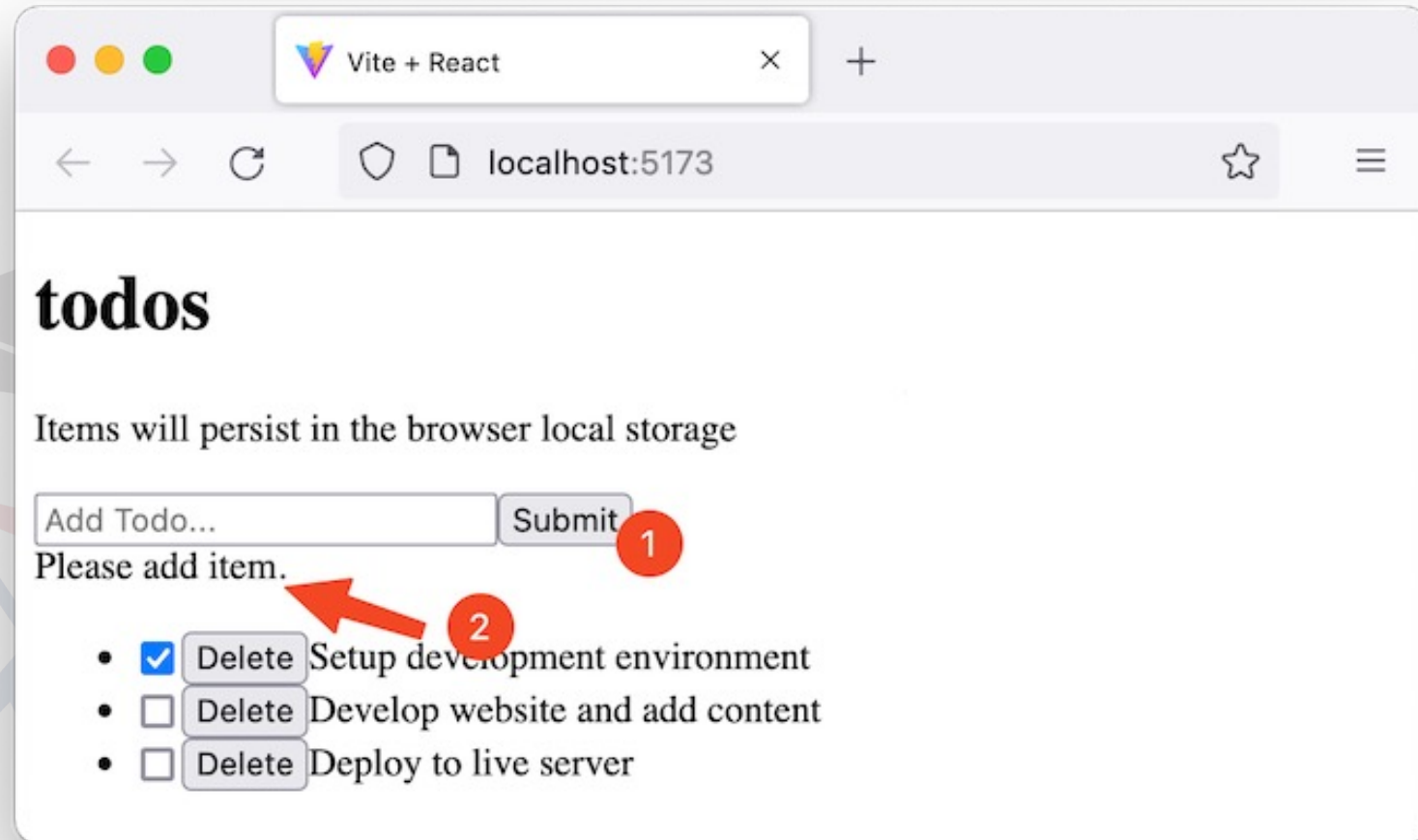


```
1  import React, { useState } from "react";
2  function Counter(props) {
3    const [count, setCount] = useState(props.initialValue);
4
5    return (
6      <div>
7        <h1>{count}</h1>
8        <button onClick={() => setCount(count + 1)}>Tambah</button>
9      </div>
10   );
11 }
12
13 function StateProps() {
14   return <Counter initialValue={10} />;
15 }
16
17 export default StateProps;
```

State & Props dalam Form Handling

Dalam React, form HTML seperti **<input>**, **<textarea>**, dan **<select>** biasanya memerlukan sedikit penanganan khusus karena mempertahankan state lokal mereka sendiri.

Untuk **mengendalikan input form**, kita menggunakan apa yang disebut "**controlled components**".



State dalam Form Handling

```
1  import React, { useState } from 'react';
2
3  function NameForm() {
4    const [name, setName] = useState('');
5
6    const handleSubmit = (event) => {
7      alert('A name was submitted: ' + name);
8      event.preventDefault();
9    };
10
11   return (
12     <form onSubmit={handleSubmit}>
13       <label>
14         Name:
15         <input type="text" value={name} onChange={e => setName(e.target.value)} />
16       </label>
17       <button type="submit">Submit</button>
18     </form>
19   );
20 }
21 export default NameForm;
```

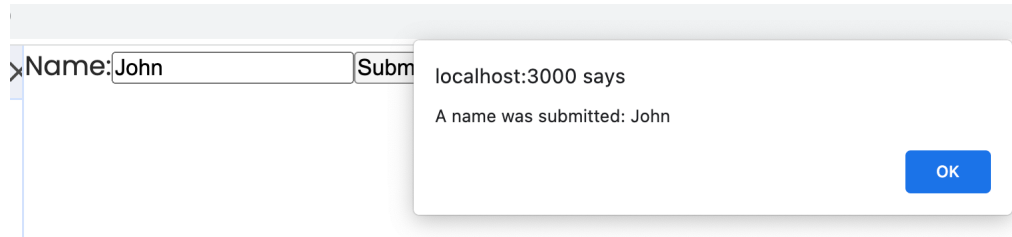
Name:

localhost:3000 says

A name was submitted: budi

OK

Props dalam Form Handling



```
1  import React, { useState } from "react";
2
3  function NameForm(props) {
4    const [name, setName] = useState(props.initialName);
5
6    const handleSubmit = (event) => {
7      props.onSubmit(name);
8      event.preventDefault();
9    };
10
11   return (
12     <form onSubmit={handleSubmit}>
13       <label>
14         Name:
15         <input
16           type="text"
17           value={name}
18           onChange={(e) => setName(e.target.value)}
19         />
20       </label>
21       <button type="submit">Submit</button>
22     </form>
23   );
24 }
25
26 function FormName() {
27   const handleNameSubmit = (name) => {
28     alert("A name was submitted: " + name);
29   };
30
31   return <NameForm initialName="John" onSubmit={handleNameSubmit} />;
32 }
33
34 export default FormName;
```

(?) Cari Tahu

- Dalam situasi apa Anda mungkin perlu menggabungkan state dan props dalam komponen?
- Bagaimana Anda akan mengatur state dan props untuk sebuah aplikasi to-do list?



