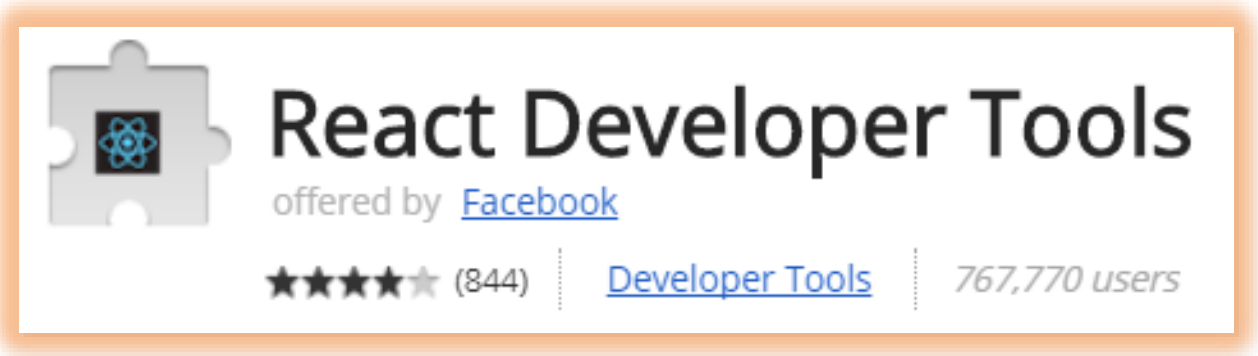
Front-End Development

React

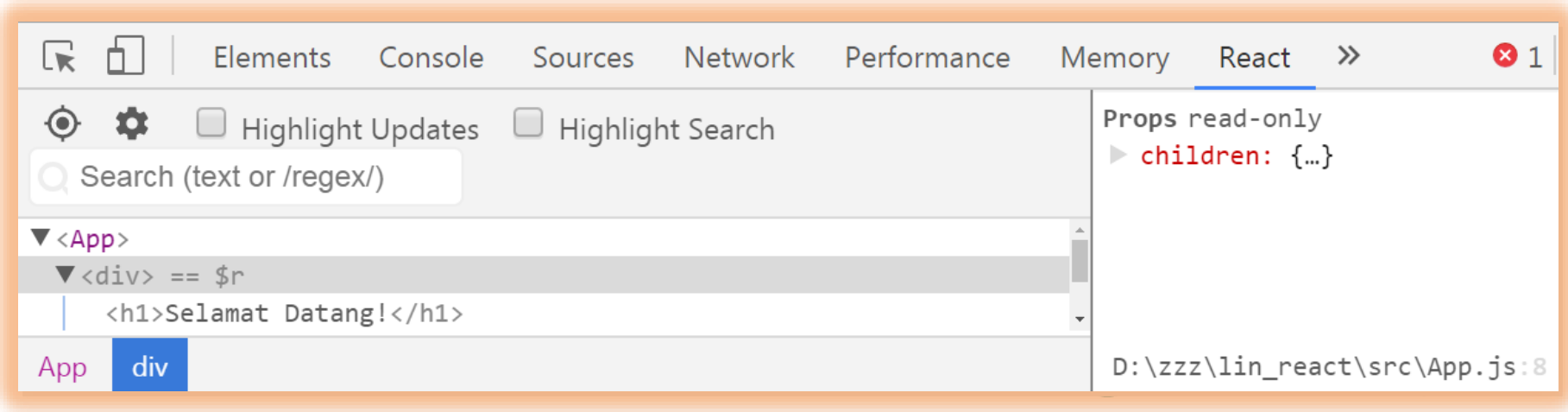
#2 Learn Once, Write Anywhere

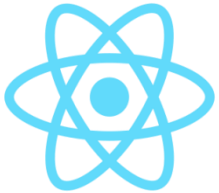
Purwadhika
Startup and Coding School

React Dev Tools



- Add React Dev Tools on Chrome from: *https://chrome.google.com/webstore/*

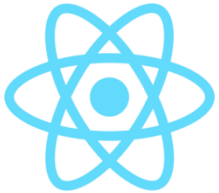




State & Props

- There are 2 types of data that control a component: *State* & *Props*.
- *Props* are set by the parent and they're fixed throughout the lifetime of a component. For data that is going to change, we have to use *State*.

Purwadhika
Startup and Coding School



State

- In general, we should initialize state in the `constructor`, and then call `setState` when we want to change it.
- State is mutable, and defines at any given time, the current state of the React component that is being rendered.

Purwadhika
Startup and Coding School

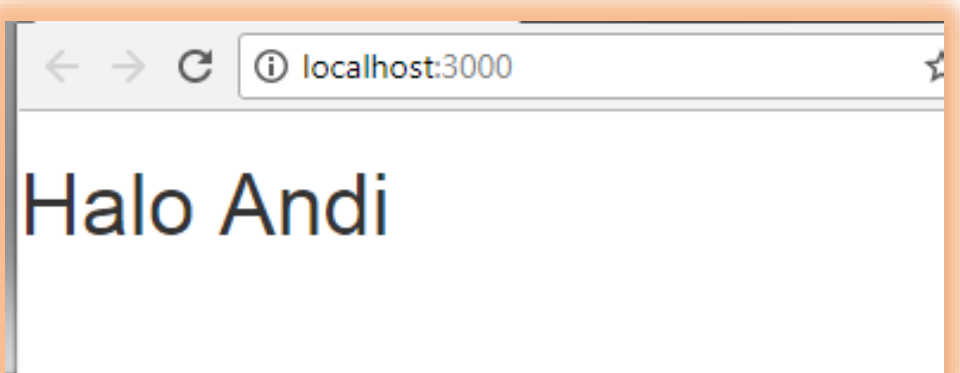
src/App.js

```
import React, { Component } from 'react';

class App extends Component {
    render() {
        return (
            <h1>Selamat datang!</h1>
        );
    }
}

export default App;
```

src/App.js

State #1

```
import React, { Component } from 'react';

class App extends Component {
    constructor(){
        super();
        this.state = {nama:'Andi'};
    }
    render() {
        return (
            <h1>Halo {this.state.nama}</h1>
        );
    }
}

export default App;
```



src/App.js

State #2

```
import React, { Component } from 'react';

class App extends Component {
    constructor(){
      super();
      this.state = {nama:'Andi', usia:21};
    }
render() {
    return (
        <div>
            <h1>Halo {this.state.nama}</h1>
            <h1>Usia {this.state.usia} th</h1>
        </div>
    );
}
}

export default App;
```

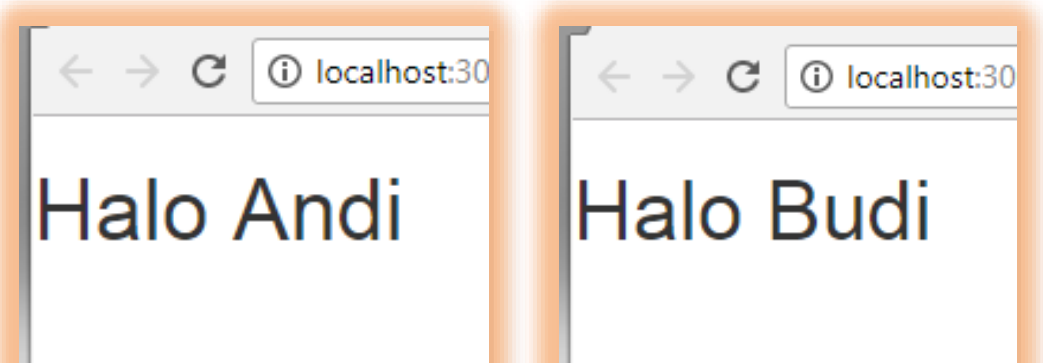
Halo Andi

Usia 21 th

src/App.js

Updating State

```
import React, { Component } from 'react';
class App extends Component {
    constructor(){
        super();
        this.state = {nama:'Andi'};
    }
    render() {
        setTimeout(() => {
            this.setState({nama: 'Budi'});
        }, 3000)
        return (
            <h1>Halo {this.state.nama}</h1>
        );
    }
}
export default App;
```



Props

- Most components can be customized with different parameters when they are created. These creation parameters are called *Props (Properties)*.
- It lets you make a component that is used in many different places in your app, with slightly different properties in each places.
- For short, props are static properties on a React component that are immutable (cannot be changed).

Purwadhika
Startup and Coding School

src/App.js

Props #1

```
import React, { Component } from 'react';
import Footer from './component/Footer';

class App extends Component {
  render() {
    var teks = "Hak cipta dilindungi";
    return (
      <div>
        <h1>Selamat datang!</h1>
        <Footer konten={teks} />
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

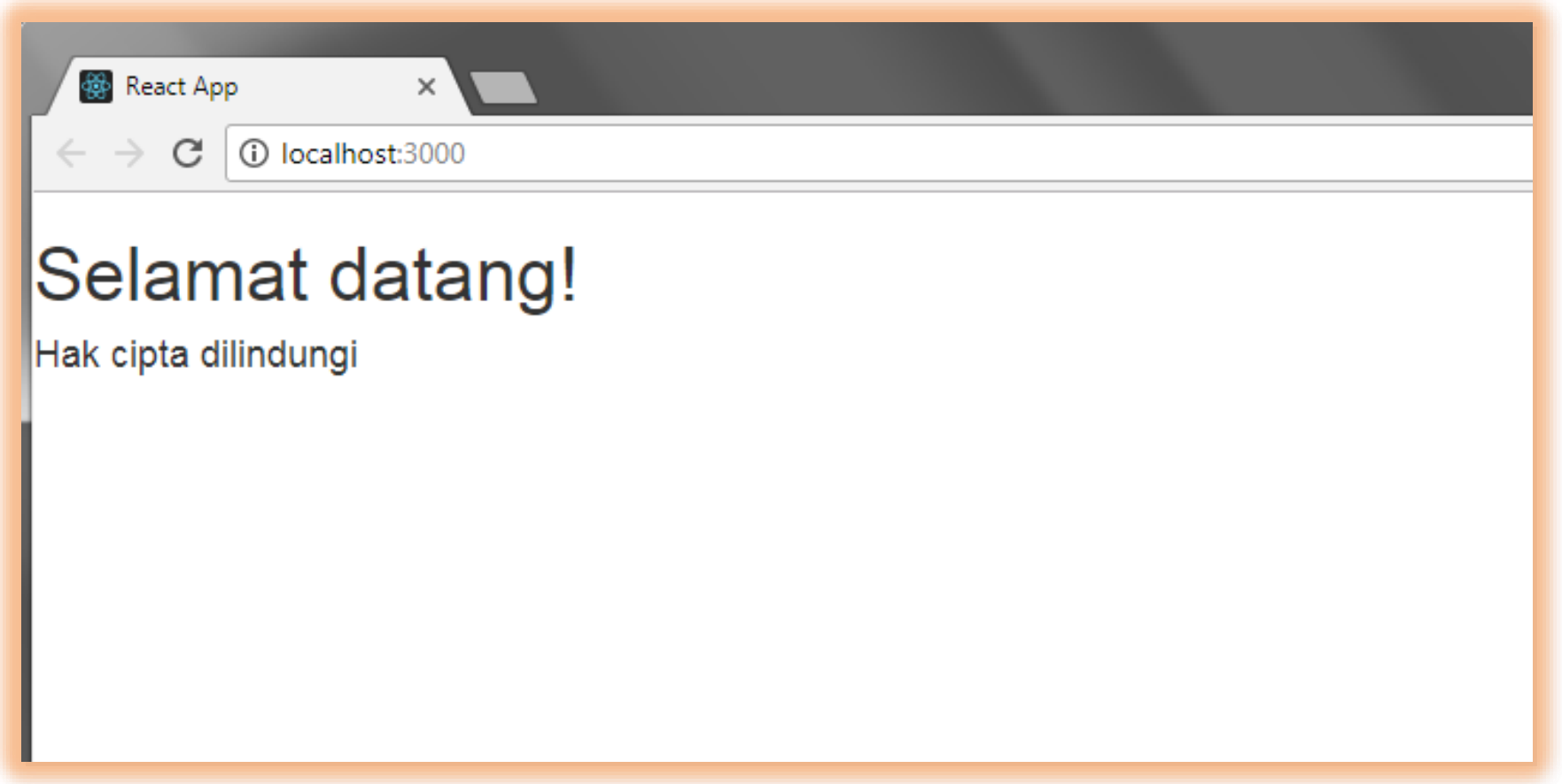
src/component/Footer.js

Props #1

```
import React, { Component } from 'react';

class Footer extends Component {
  render() {
    return (
      <div>
        <h4>{this.props.konten}</h4>
      </div>
    );
  }
}
export default Footer;
```

Purwadhika
Startup and Coding School

Props #1



Purwadhika
Startup and Coding School

src/App.js

Props #2

```
import React, { Component } from 'react';
import Footer from './component/Footer';

class App extends Component {
  render() {
    var andi = {nama:"Andi",usia:24};
    return (
      <div>
        <h1>Selamat datang!</h1>
        <Footer id={andi.nama} umur={andi.usia}/>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

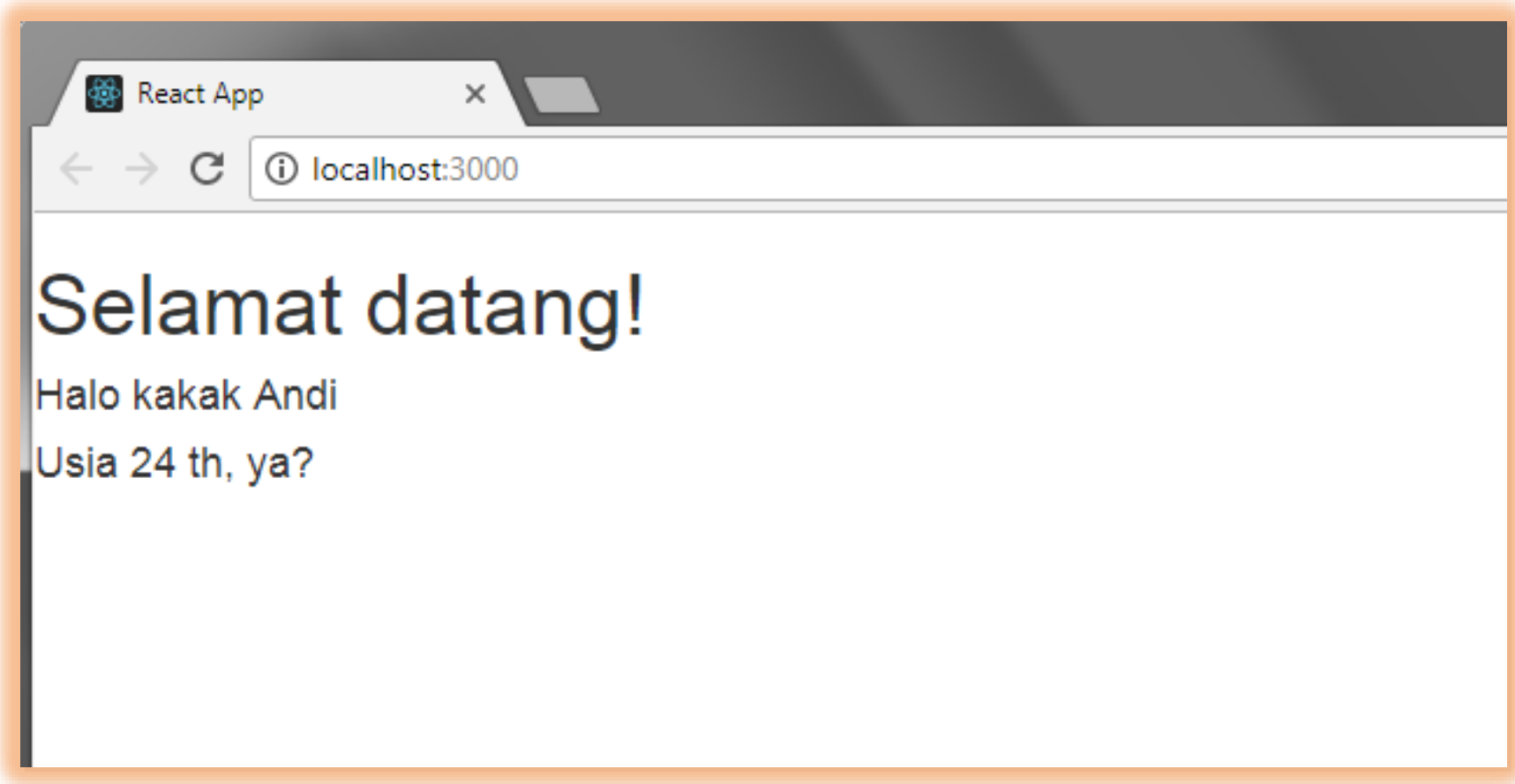
src/component/Footer.js

Props #2

```
import React, { Component } from 'react';

class Footer extends Component {
  render() {
    return (
      <div>
        <h4>Halo kakak {this.props.id}</h4>
        <h4>Usia {this.props.umur} th, ya?</h4>
      </div>
    );
  }
}
export default Footer;
```

Purwadhika
Startup and Coding School

Props #2



Purwadhika
Startup and Coding School

src/App.js

State → Props

```
import React, { Component } from 'react';
import Footer from './component/Footer';

class App extends Component {
  constructor(){
    super();
    this.state = {nama:'Andi'};
  }
  render() {
    return (
      <div>
        <Footer id={this.state.nama} />
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

src/component/Footer.js

State → Props

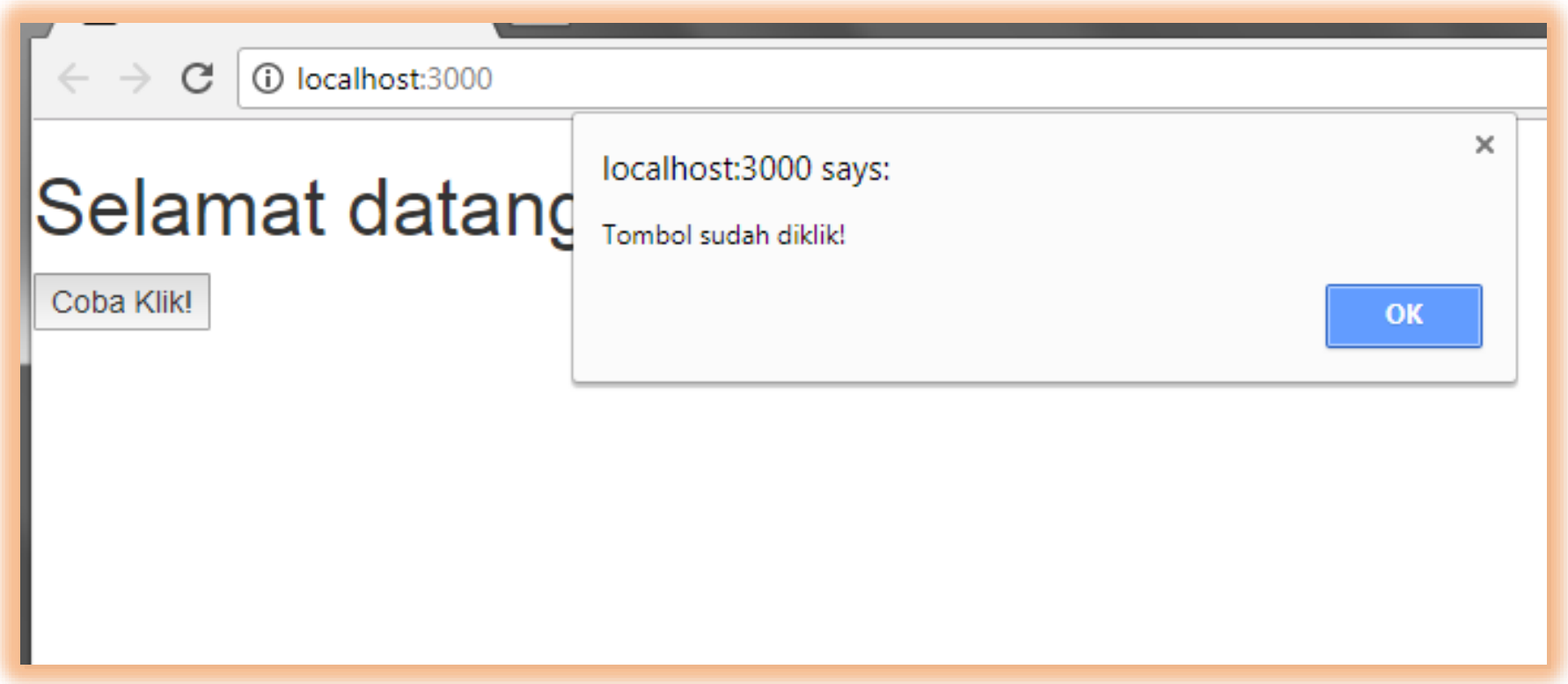
```
import React, { Component } from 'react';

class Footer extends Component {
  render() {
    return (
      <div>
        <h1>Halo {this.props.id}</h1>
      </div>
    );
  }
}
export default Footer;
```

Purwadhika
Startup and Coding School

Handling Events

Purwadhika
Startup and Coding School

Click Event



Purwadhika
Startup and Coding School

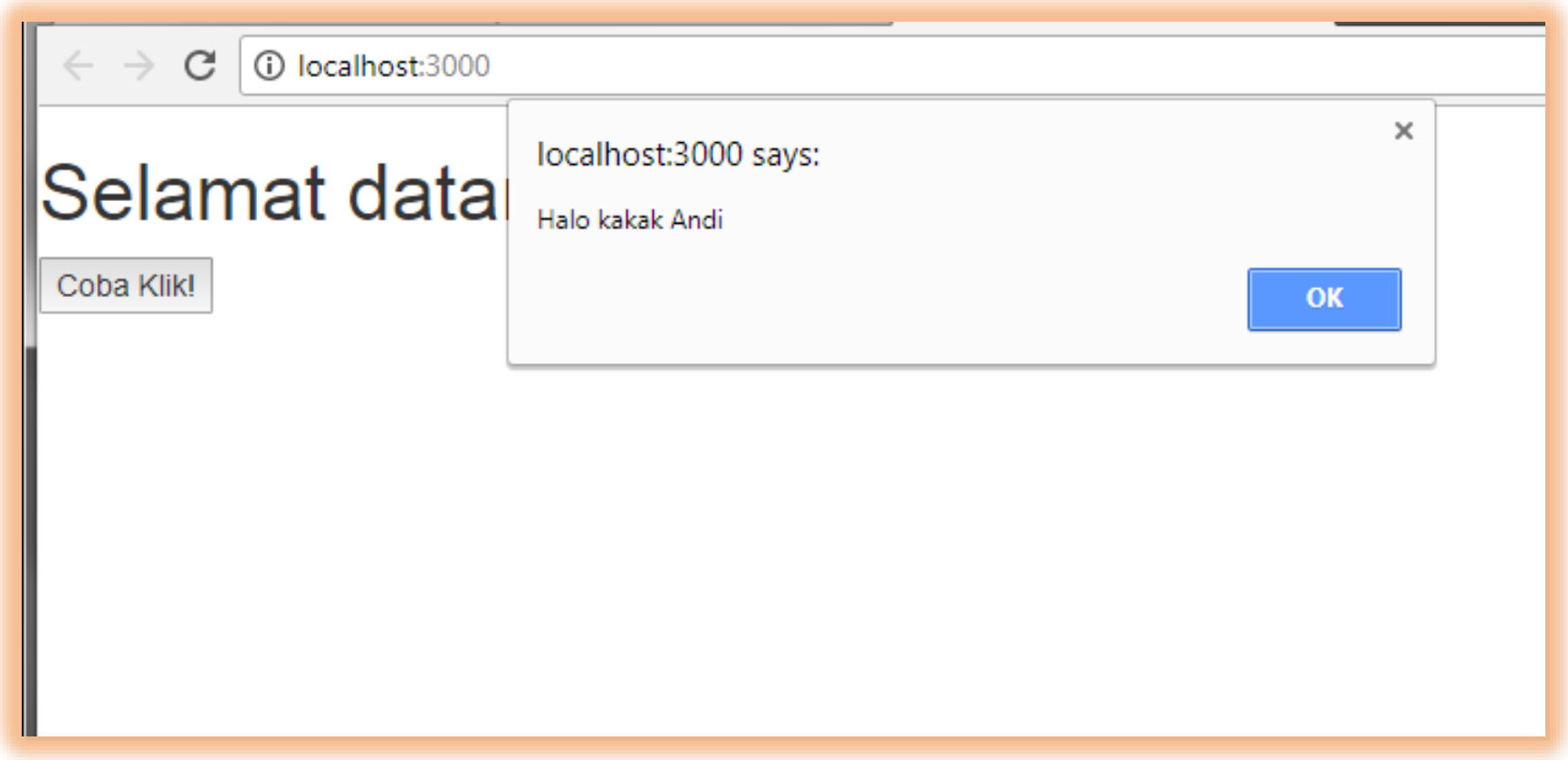
src/App.js

Click Event

```
import React, { Component } from 'react';
class App extends Component {
  klik(){
    alert('Tombol sudah diklik!');
  }
  render() {
    return (
      <div>
        <h1>Selamat datang!</h1>
        <button onClick={this.klik}>
          Coba Klik!
        </button>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

Click Event *with param*



Purwadhika
Startup and Coding School

src/App.js

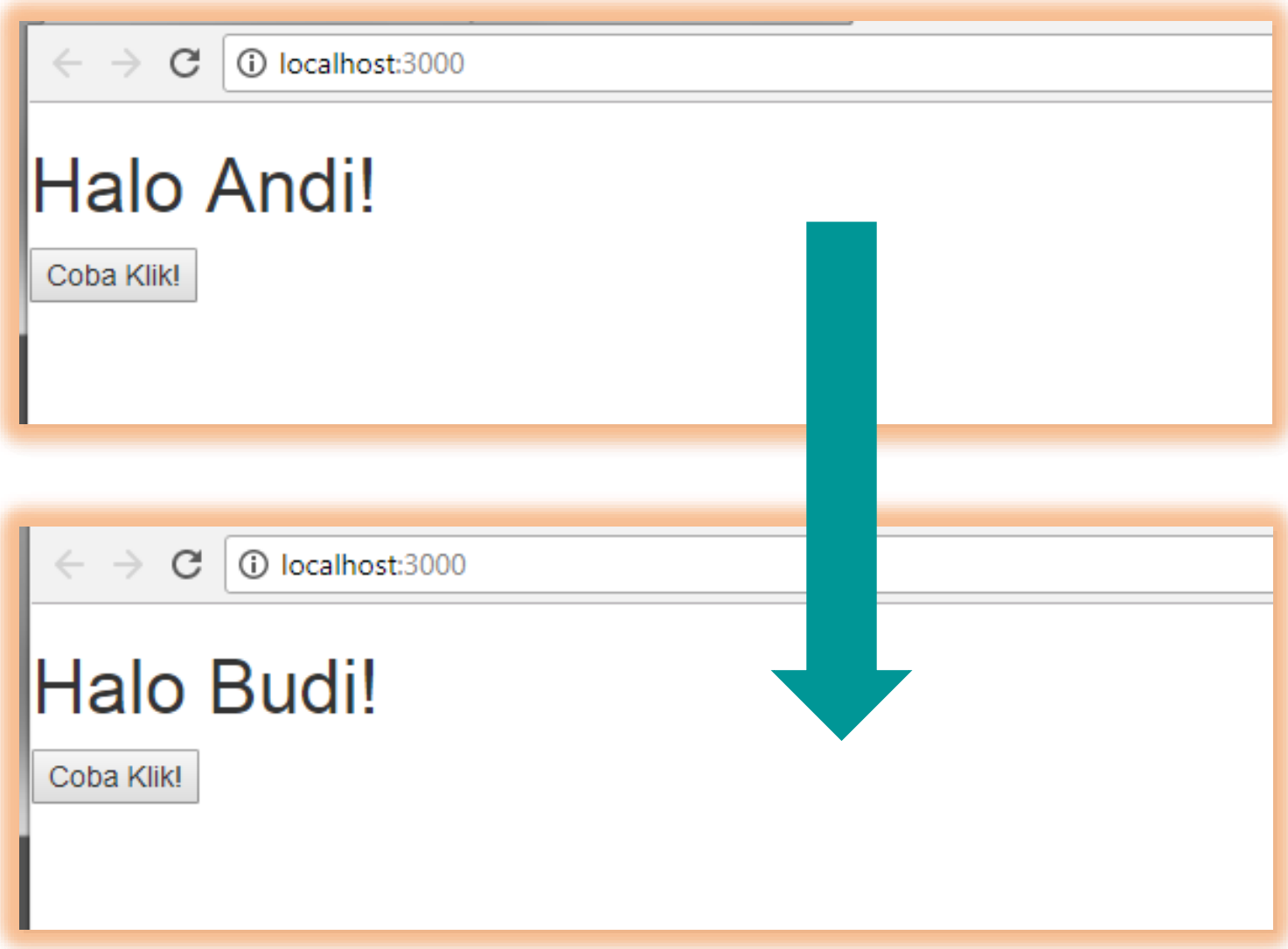
Click Event *with param*

```
import React, { Component } from 'react';
class App extends Component {
  klik(siapa){
    alert('Halo kakak '+siapa);
  }
  render() {
    return (
      <div>
        <h1>Selamat datang!</h1>
        <button onClick={()=>{this.klik('Andi');}}>
          Coba Klik!
        </button>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

Click Event

update State #1



src/App.js

Click Event

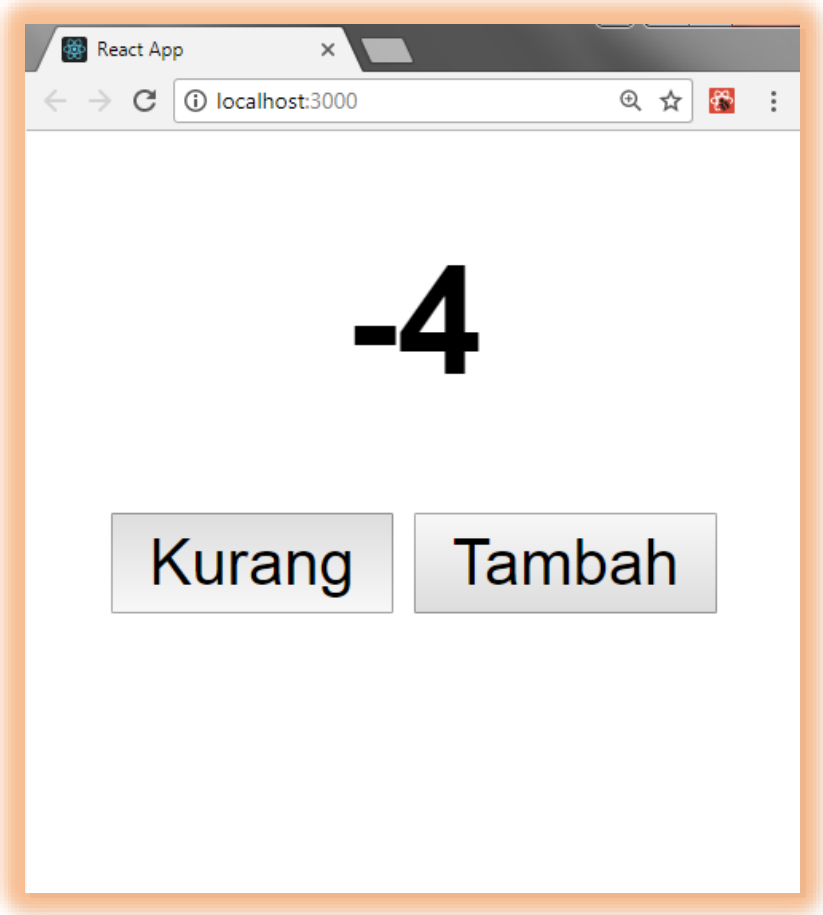
update State #1

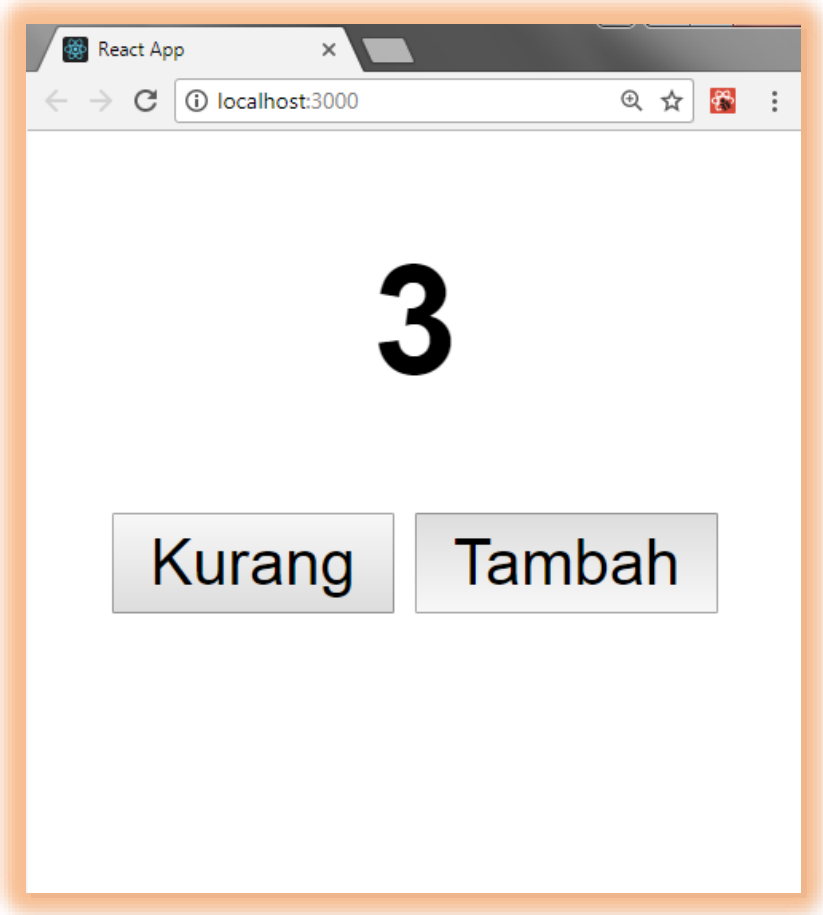
```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Andi'};
  }
  klik(siapa){
    this.setState({user:siapa});
  }
  render() {
    return (
      <div>
        <h1>Halo {this.state.user}!</h1>
        <button onClick={()=>{this.klik('Budi');}}>
          Coba Klik!
        </button>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

Click Event

update State #2





Purwadhika
Startup and Coding School

src/App.js

Click Event

update State #2

Part 1

```
import React, { Component } from 'react';

class App extends Component {
  state = { count: 0 }

  tambah = () => {
    this.setState({
      count: this.state.count + 1
    });
  }

  kurang = () => {
    this.setState({
      count: this.state.count - 1
    });
  }
```

Purwadhika
Startup and Coding School

src/App.js

Click Event

update State #2

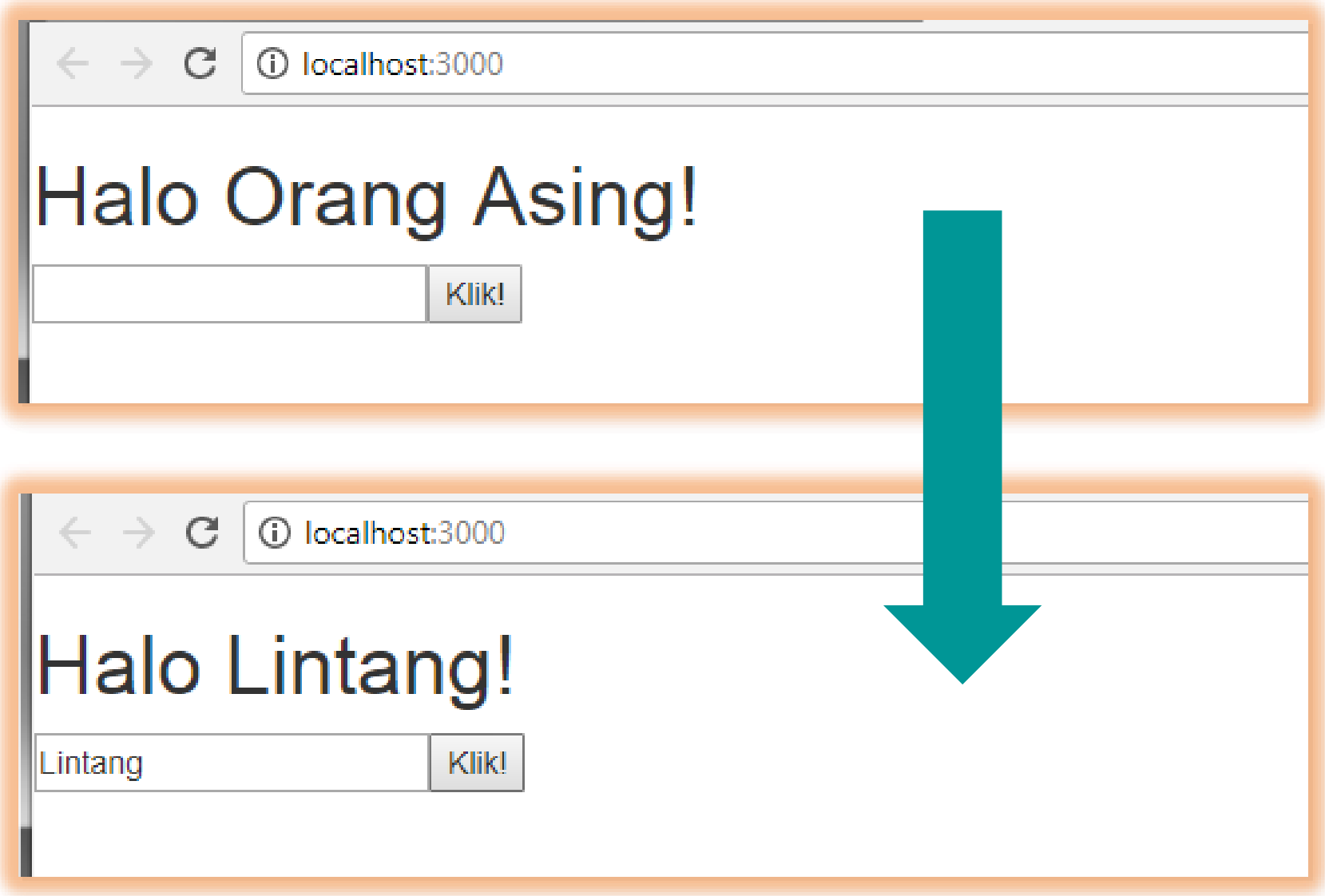
Part 2

```
render(){
  return (
    <div>
    <center>
    <h1>{this.state.count}</h1>
    <div>
    <button onClick = {this.kurang}>Kurang</button>
    <span> </span>
    <button onClick = {this.tambah}>Tambah</button>
    </div>
    </center>
    </div>
  );
}
}

export default App;
```

Purwadhika
Startup and Coding School

Click Event

update State from user input



Click Event

update State from user input

src/App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Orang Asing'};
  }
  klik(){
    this.setState({user: this.refs.nama.value});
  }
  render() {
    return (
      <div>
        <h1>Halo {this.state.user}!</h1>
        <input ref="nama" type="text"/>
        <button onClick={()=>{this.klik();}}>Klik!</button>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

Events

- **Keyboard Events**
  onKeyDown onKeyPress onKeyUp

- **Focus Events**
  onFocus onBlur

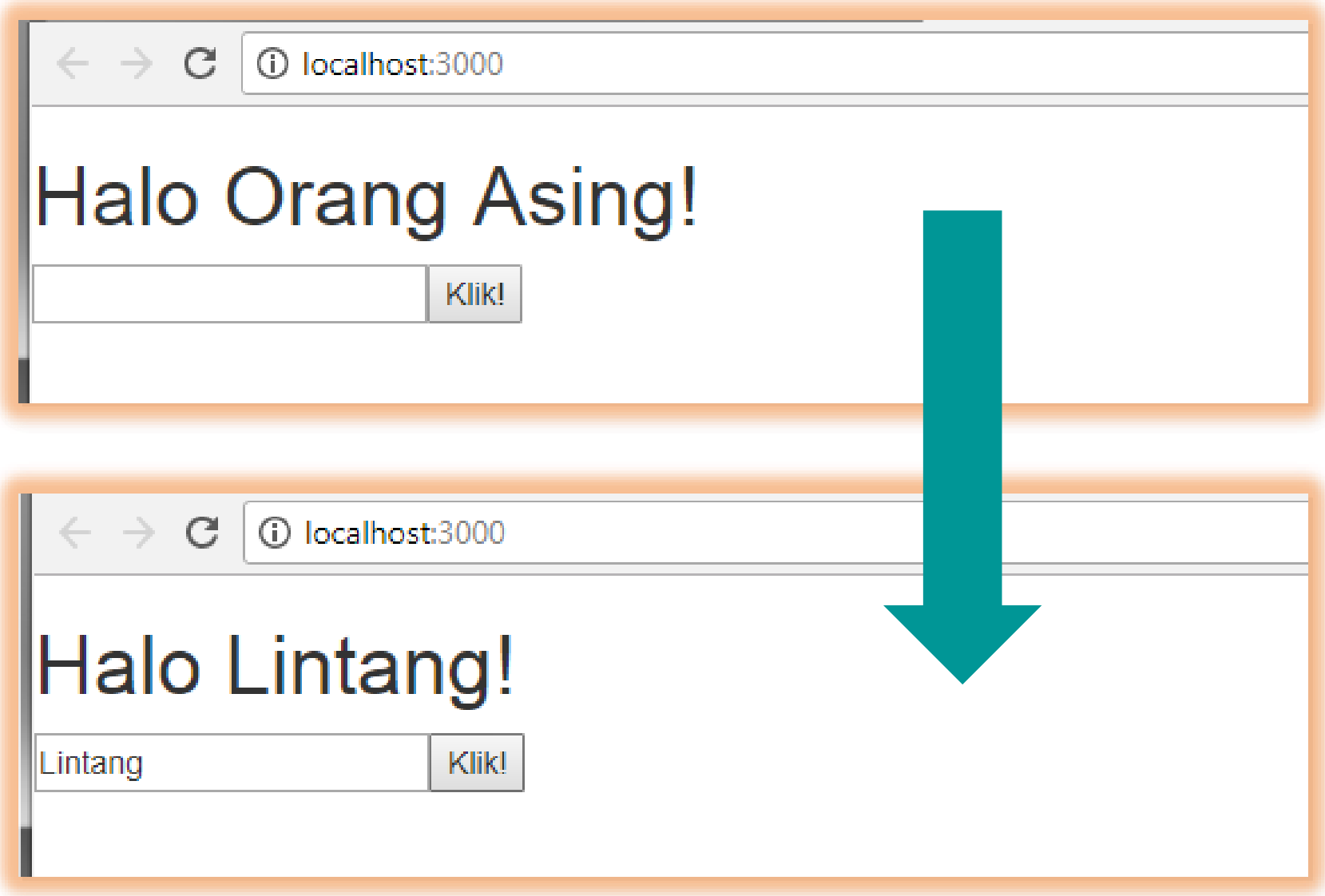
- **Form Events**
  onChange onInput onInvalid onSubmit

- **Mouse Events**
  onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave onMouseMove onMouseOut onMouseOver onMouseUp

Purwadhika
Startup and Coding School

Double Click Event

update State from user input



src/App.js

Double Click

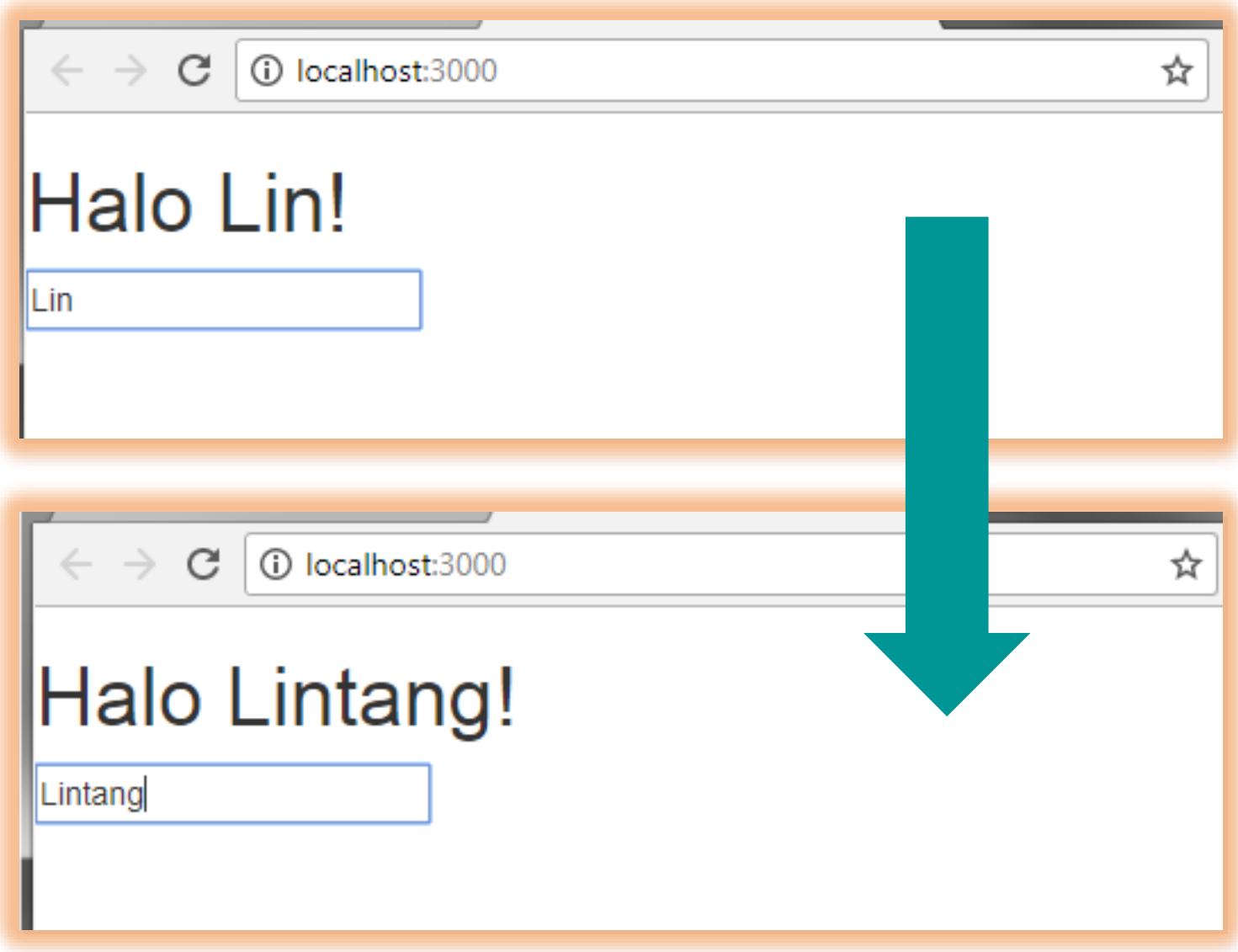
update State from user input

```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Orang Asing'};
  }
  klik(){
    this.setState({user: this.refs.nama.value});
  }
  render() {
    return (
      <div>
        <h1>Halo {this.state.user}!</h1>
        <input ref="nama" type="text"/>
        <button onDoubleClick={()=>{this.klik();}}>
          Klik!
        </button>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

onInput Event

update State from user input



onInput Event

update State from user input

src/App.js

```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Orang Asing'};
  }
  klik(){
    this.setState({user: this.refs.nama.value});
  }
  render() {
    return (
      <div>
        <h1>Halo {this.state.user}!</h1>
        <input ref="nama" type="text"
         onInput={()=>{this.klik();}}/>
      </div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

Life-cycle Methods

#1 Mounting

- componentWillMount(){}

Invoked once on both client and server, immediately before the initial rendering occurs.

- componentDidMount(){}

Invoked once only on client, immediately after the initial rendering occurs.

Purwadhika
Startup and Coding School

Life-cycle Methods

Mounting

src/App.js

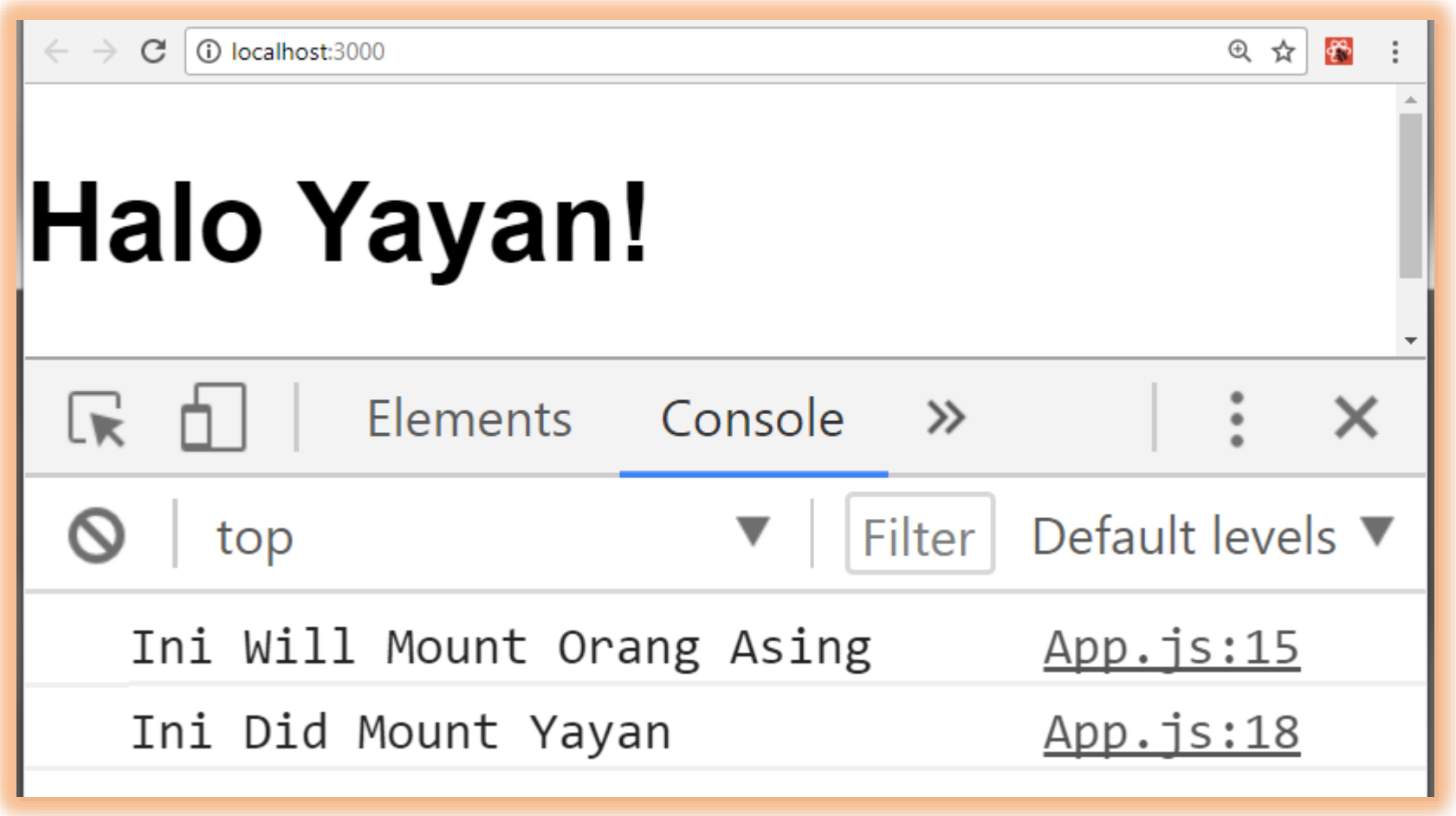
```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Orang Asing'};
  }
  componentWillMount(){
    this.setState({user:'Yayan'});
    console.log('Ini Will Mount '+this.state.user);
  }
  componentDidMount(){
    console.log('Ini Did Mount '+this.state.user);
  }
  render() {
    return (
      <div><h1>Halo {this.state.user}!</h1></div>
    );
  }
}
export default App;
```

Purwadhika
Startup and Coding School

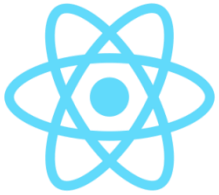
src/App.js

Life-cycle Methods

Mounting



Purwadhika
Startup and Coding School



Life-cycle Methods

#2 Updating

- componentWillUpdate()

Invoked immediately before rendering when new props or state are being received. This isn't called for the initial render.

- componentDidUpdate()

Invoked immediately after the component's updates are flushed to the DOM. This isn't called for the initial render.

Purwadhika
Startup and Coding School

Life-cycle
Updating

src/App.js

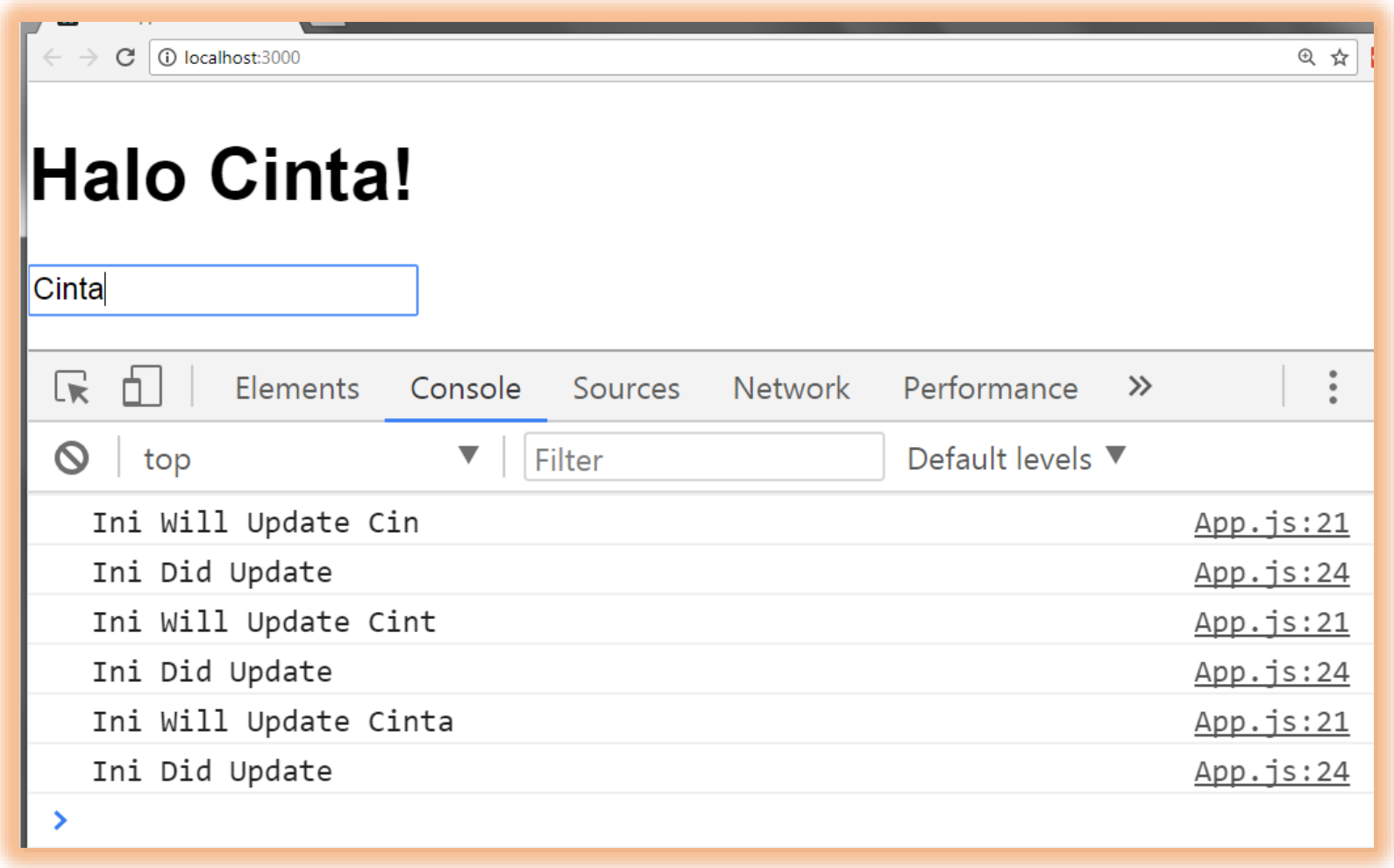
```
import React, { Component } from 'react';
class App extends Component {
  constructor(){
    super();
    this.state = {user:'Orang Asing'};
  }
  klik(){
    this.setState({user: this.refs.nama.value});
  }
  componentWillUpdate(x, y){
    console.log('Ini Will Update '+y.user);
  }
  componentDidUpdate(){
    console.log('Ini Did Update ');
  }
  render() {
    return (
 <div><h1>Halo {this.state.user}!</h1>
 <input ref="nama" type="text" onInput={()=>{this.klik();}}/>
 </div>
  );}}
export default App;
```

Purwadhika
Startup and Coding School

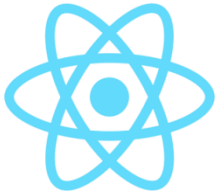
src/App.js

Life-cycle Methods

Updating



Purwadhika
Startup and Coding School



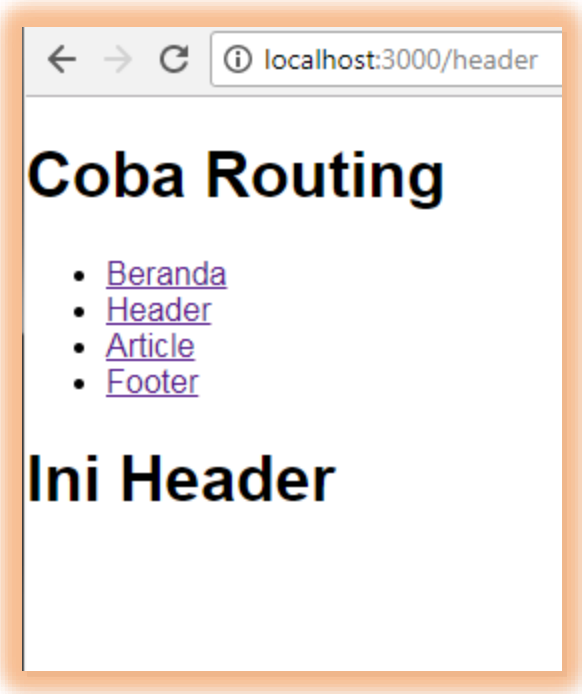
Router

- A router allows an application to navigate between different components, changing the browser URL, modifying the browser history, and keeping the UI state in sync.
- React focuses only on building user interfaces, so it doesn't have a built-in solution for routing.
- We can use ***React Router DOM***, which allows us to define routes on our web app. First, install this package on our React project directory:

```
$ npm install react-router-dom --save
```

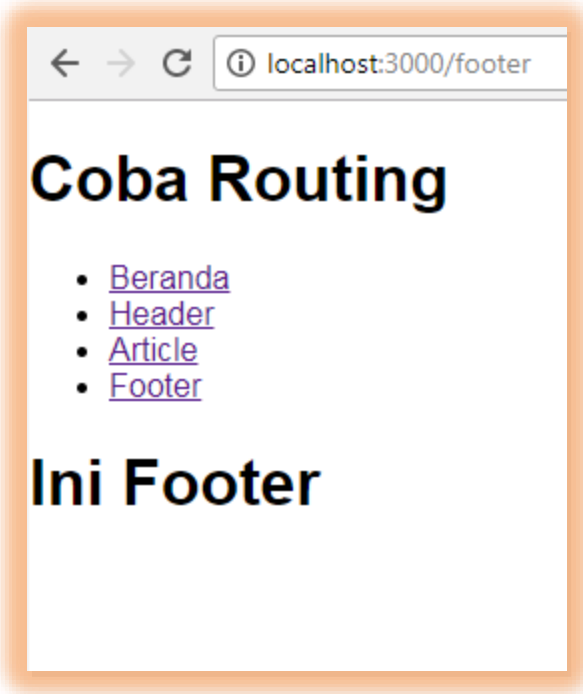
Purwadhika
Startup and Coding School

Routing



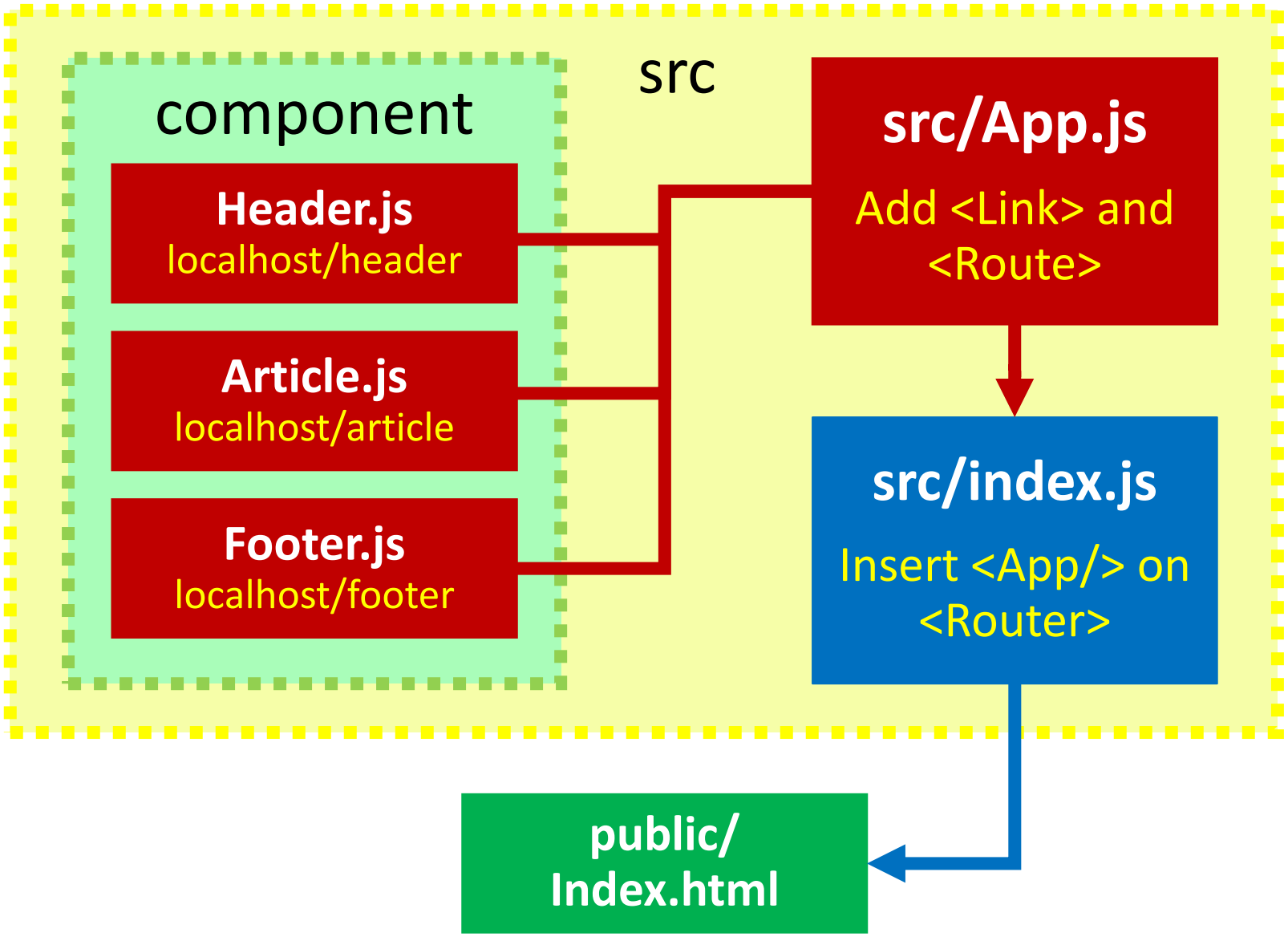
Header.js
localhost/header

Article.js
localhost/article



Footer.js
localhost/footer

Purwadhika
Startup and Coding School



Purwadhika
Startup and Coding School

src/component/Header.js

#1 component

```
import React, { Component } from 'react';

class Header extends Component {
    render() {
        return (
            <h1>Ini Header!</h1>
        );
    }
}

export default Header;
```

Purwadhika
Startup and Coding School

src/component/Article.js

#2 component

```
import React, { Component } from 'react';

class Article extends Component {
    render() {
        return (
            <h1>Ini Article!</h1>
        );
    }
}

export default Article;
```

Purwadhika
Startup and Coding School

src/component/Footer.js

#3 component

```
import React, { Component } from 'react';

class Footer extends Component {
    render() {
        return (
            <h1>Ini Footer!</h1>
        );
    }
}

export default Footer;
```

Purwadhika
Startup and Coding School

src/index.js

Insert Router!

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import registerServiceWorker from
'./registerServiceWorker';

import { BrowserRouter } from 'react-router-dom';

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
document.getElementById('root'));

registerServiceWorker();
```

Purwadhika
Startup and Coding School

src/App.js

Part 1 Import

```
import React, { Component } from 'react';
import { Link, Route } from 'react-router-dom';

import Header from './component/Header';
import Footer from './component/Footer';
import Article from './component/Article';
```

Purwadhika
Startup and Coding School

src/App.js

Part 2 Insert Link

```
class App extends Component {
render() {
return (
<div>
<h1>Coba Routing</h1>
<ul>
<li><Link to="/">Beranda</Link></li>
<li><Link to="/header">Header</Link></li>
<li><Link to="/article">Article</Link></li>
<li><Link to="/footer">Footer</Link></li>
</ul>
```

Purwadhika
Startup and Coding School

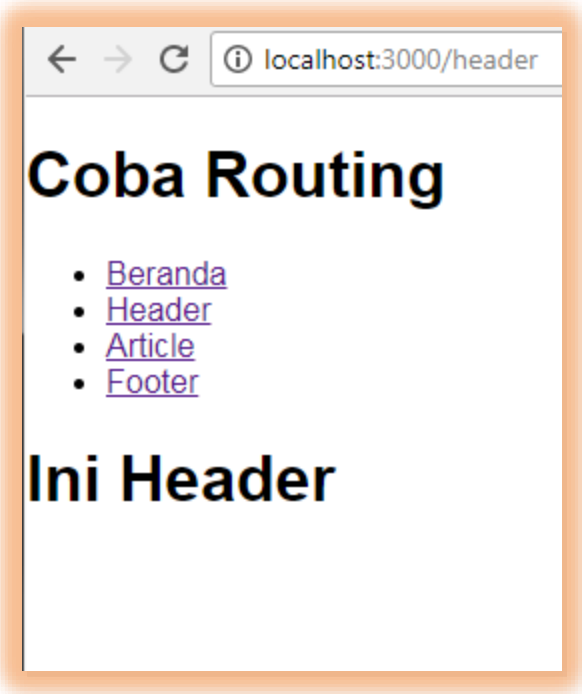
src/App.js

Part 3 Insert Route

```
<div>
<Route exact path="/" component={'Home'}/>
<Route path="/article" component={Article}/>
<Route path="/header" component={Header}/>
<Route path="/footer" component={Footer}/>
</div>
</div>
);
}
}
export default App;
```

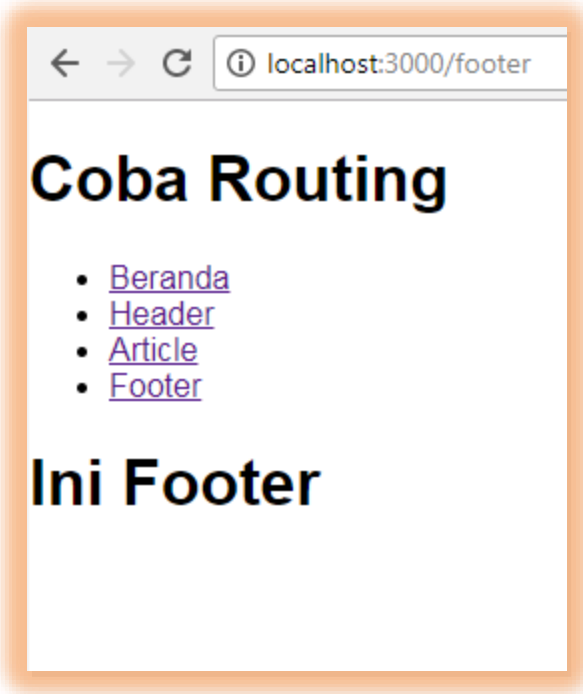
Purwadhika
Startup and Coding School

Routing



Header.js
localhost/header

Article.js
localhost/article



Footer.js
localhost/footer

Purwadhika
Startup and Coding School

Front-End Development

React

#2 Learn Once, Write Anywhere

Purwadhika
Startup and Coding School