Pemrograman Web Modern

Insan Taufik, S.Kom., M.Kom.

Komponen Dasar (Fungsional)

Sub-Materi

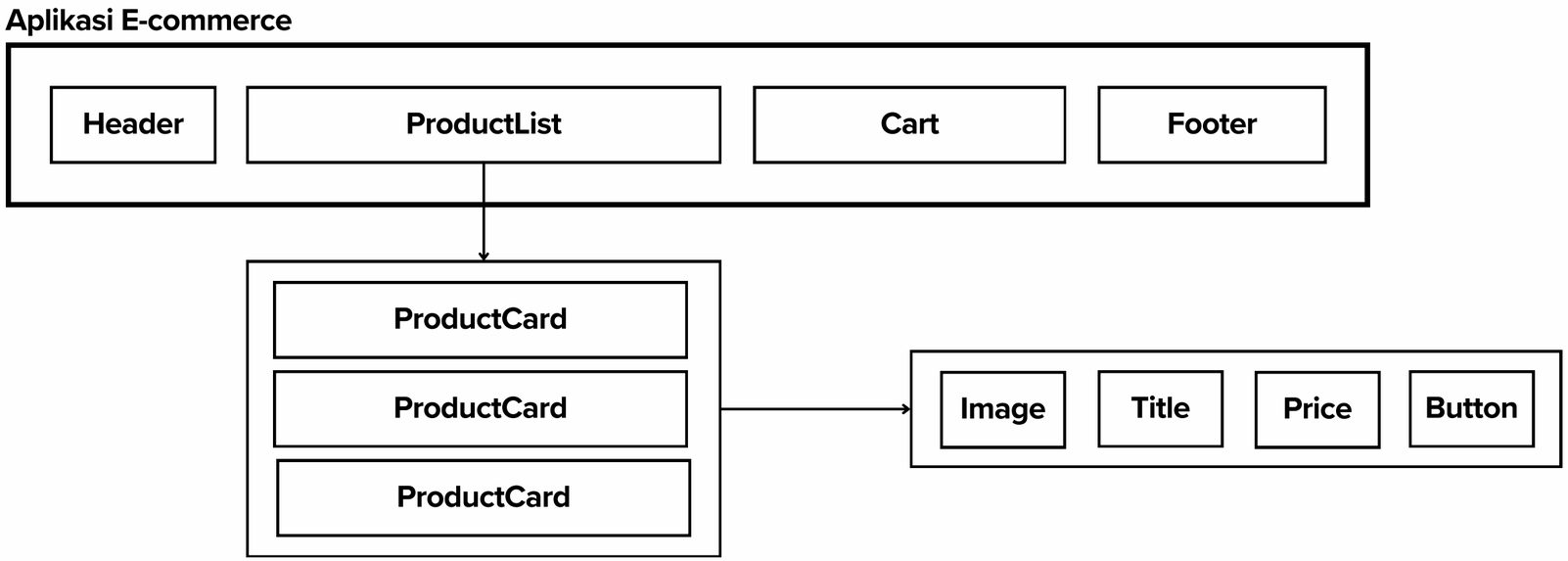
Konsep Komponen sebagai pondasi UI. Membuat komponen fungsional. Menerapkan import dan export komponen. Menentukan dan menggunakan Props (Properties) untuk komunikasi data antar komponen (satu arah).

Tujuan Pembelajaran (Capaian)

Mahasiswa dapat memecah UI menjadi komponen-komponen yang lebih kecil dan menggunakan Props untuk membuat komponen reusable.

Apa itu Komponen?

- Blok pembangun UI yang independen dan reusable
- Seperti fungsi JavaScript yang mengembalikan elemen React
- Philosophy: "Divide and Conquer" - pecah UI kompleks menjadi bagian kecil



Jenis Komponen React

1. Komponen Fungsional (Function Components)

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

2. Komponen Klasik (Class Components)

```jsx
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Membuat Komponen Fungsional

Struktur Dasar:

```jsx
// 1. Function Declaration
function ComponentName() {
  return <div>Content</div>;
}

// 2. Arrow Function (lebih modern)
const ComponentName = () => {
  return <div>Content</div>;
};

// 3. Implicit Return
const ComponentName = () => <div>Content</div>;
```

Aturan Penamaan:

- PascalCase (huruf pertama kapital)
- Deskriptif tentang fungsinya
- Contoh: Button, UserProfile, ProductList

Import dan Export Komponen

Export

```
// Named Export (banyak komponen dalam satu file)
export function Button() { /* ... */ }
export function Card() { /* ... */ }

// Default Export (satu komponen utama per file)
function Header() { /* ... */ }
export default Header;
```

Import

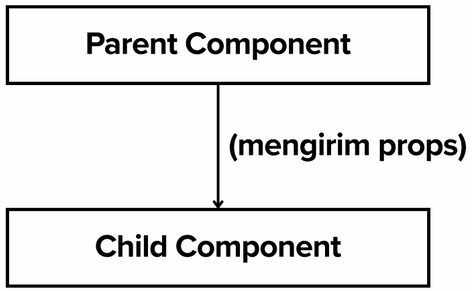
```
// Import Named Export
import { Button, Card } from './components';

// Import Default Export
import Header from './components/Header';

// Import dengan alias
import { Button as Tombol } from './components';
```

Pengenalan Props (Properties)

Apa itu Props?

- Data yang diterima komponen dari parent component
- Seperti parameter fungsi
- Read-only (tidak bisa diubah oleh komponen itu sendiri)
- Hanya bisa mengirim 1 props, jadi tidak bisa mengirim lebih dari 1 props misalnya function x(prop1, prop2)

Cara Kerja:




```jsx
// Parent mengirim props
<UserCard name="John" age={25} isVerified={true} />

// Child menerima props
const UserCard = (props) => {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
      {props.isVerified && <span>✓ Verified</span>}
    </div>
  );
};
```

Destructuring Props

Destructuring Props di React.js adalah teknik untuk mengambil (memecah) nilai-nilai tertentu dari props agar lebih mudah digunakan dalam komponen. Ini memanfaatkan fitur object destructuring dari JavaScript.

Tanpa Destructuring:

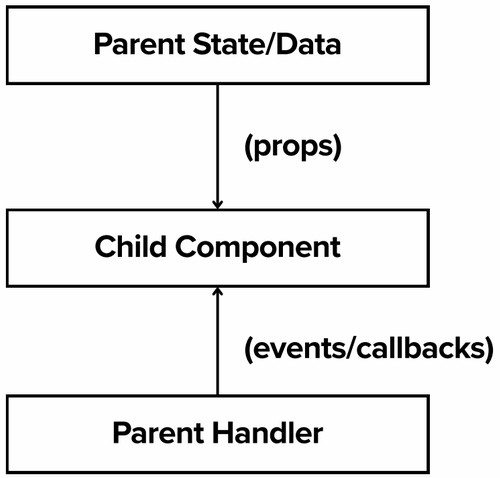
```jsx
const UserCard = (props) => {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
    </div>
  );
};
```

Dengan Destructuring (lebih clean):

```jsx
// Destructuring di parameter
const UserCard = ({ name, age, isVerified }) => {
  return (
    <div>
      <h2>{name}</h2>
      <p>Age: {age}</p>
      {isVerified && <span>✓ Verified</span>}
    </div>
  );
};

// Destructuring di function body
const UserCard = (props) => {
  const { name, age, isVerified } = props;
  return /* ... */;
};
```

One-Way Data Flow

Konsep Data Satu Arah:



Karakteristik:

- Data mengalir dari parent ke child melalui props
- Child tidak bisa mengubah props langsung
- Child bisa memicu perubahan melalui callback functions

Keuntungan Komponen dan Props

Reusability:

- Satu komponen, banyak penggunaan
- Contoh: <Button> untuk berbagai aksi

Maintainability:

- Kode lebih terorganisir
- Mudah debug dan test

Scalability:

- Mudah dikembangkan
- Tim development lebih efisien

Ringkasan

Yang telah dipelajari:

- ✓ Konsep komponen sebagai pondasi UI
- ✓ Membuat komponen fungsional
- ✓ Import/export komponen
- ✓ Penggunaan Props untuk komunikasi data
- ✓ One-way data flow