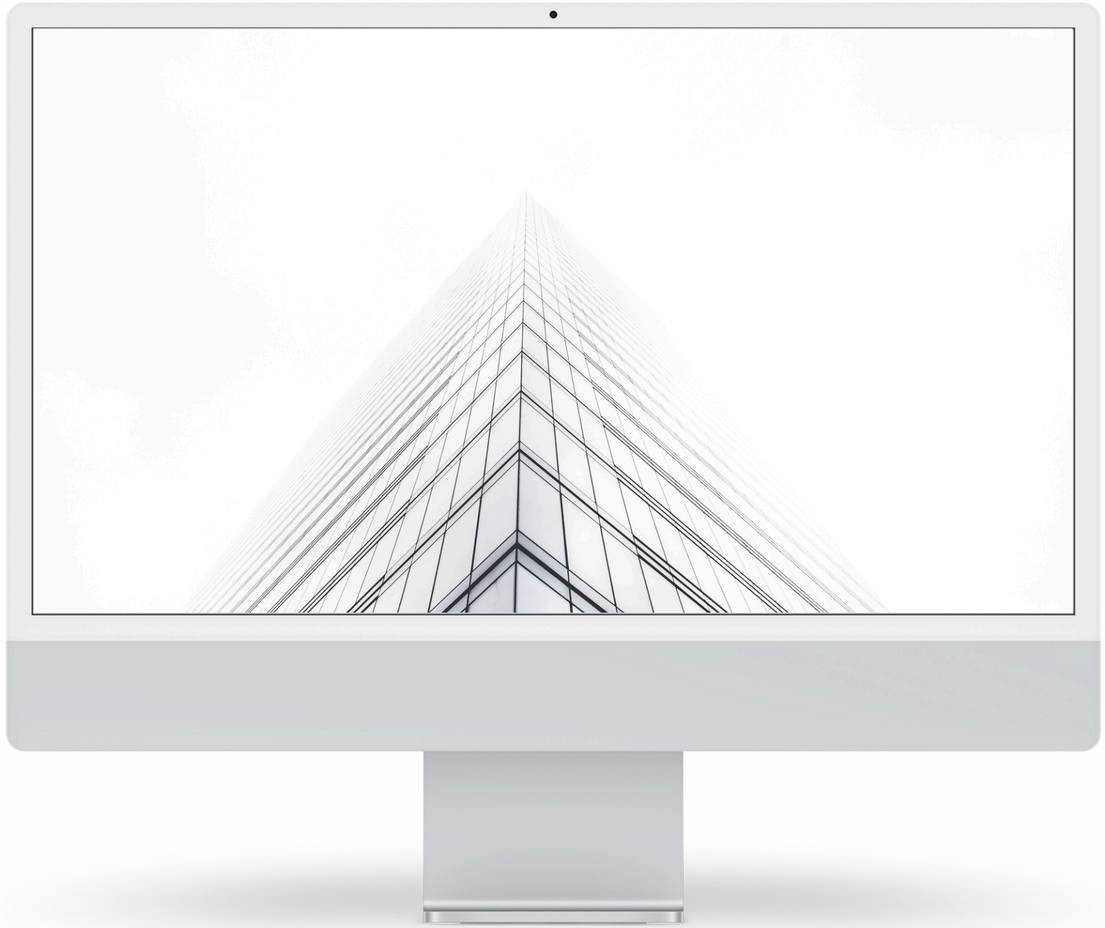
Pemrograman Web Modern

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



Styling pada React (Modern Approach)

Sub-Materi

Berbagai metode styling di React: CSS biasa, CSS Modules, Styled-Components atau Tailwind CSS. Implementasi styling kondisional. Menggunakan library komponen UI (misalnya Material UI / Ant Design).

Tujuan Pembelajaran (Capaian)

Mampu menerapkan gaya yang terstruktur, modular, dan modern pada aplikasi React.

Agenda

1. Pengenalan Styling di React
2. CSS Biasa
3. CSS Modules
4. Styled-Components
5. Tailwind CSS
6. Styling Kondisional
7. Library Komponen UI (Material UI/Ant Design)

Pengenalan Styling di React

```jsx
// Komponen React dengan styling dasar
function MyComponent() {
  return (
    <div style={{ color: 'blue', fontSize: '16px' }}>
      Hello World!
    </div>
  );
}
```

Penjelasan:

- React mendukung berbagai pendekatan styling
- Dapat menggunakan inline styles dengan objek JavaScript
- Setiap properti CSS ditulis dalam camelCase

CSS Biasa

```jsx
// Button.jsx
import './Button.css';

function Button() {
  return <button className="my-button">Click Me</button>;
}
```

```css
/* Button.css */
.my-button {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
}
```

Penjelasan:

- Import file CSS seperti biasa
- Gunakan className instead of class
- Global scope - bisa terjadi konflik nama class

CSS Modules

```css
// Button.module.css
.button {
  background-color: #28a745;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
}
```

```jsx
// Button.jsx
import styles from './Button.module.css';

function Button() {
  return <button className={styles.button}>Click Me</button>;
}
```

Penjelasan:

- CSS Modules memberikan scope lokal
- Nama class di-generate secara unik
- Mencegah konflik nama class

Styled-Components

```
import styled from 'styled-components';

const StyledButton = styled.button`
  background-color: ${props => props.primary ? '#007bff' : '#6c757d'};
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 4px;
  cursor: pointer;

  &:hover {
    opacity: 0.8;
  }
`;

function Button({ primary }) {
  return <StyledButton primary={primary}>Click Me</StyledButton>;
}
```

Penjelasan:

- CSS-in-JS approach
- Komponen dan styling dalam satu file
- Dapat menerima props untuk styling dinamis

Tailwind CSS

```jsx
function Button({ variant = 'primary' }) {
  const baseClasses = "px-4 py-2 rounded font-semibold focus:outline-none";
  const variantClasses = {
    primary: "bg-blue-500 text-white hover:bg-blue-600",
    secondary: "bg-gray-500 text-white hover:bg-gray-600"
  };

  return (
    <button className={`${baseClasses} ${variantClasses[variant]}`}>
      Click Me
    </button>
  );
}
```

Penjelasan:

- Utility-first CSS framework
- Kelas yang telah didefinisikan sebelumnya
- Sangat cepat untuk development

Styling Kondisional

```jsx
function Alert({ type, message }) {
  const getStyle = () => {
    switch(type) {
      case 'success':
        return { backgroundColor: '#d4edda', color: '#155724' };
      case 'error':
        return { backgroundColor: '#f8d7da', color: '#721c24' };
      default:
        return { backgroundColor: '#d1ecf1', color: '#0c5460' };
    }
  };

  return (
    <div style={getStyle()} className="p-3 rounded mb-3">
      {message}
    </div>
  );
}
```

Penjelasan:

- Styling berdasarkan state atau props
- Dapat menggunakan ternary operator atau fungsi
- Fleksibel untuk UI yang dinamis

Library Komponen UI - Material UI

```jsx
import { Button, Card, CardContent } from '@mui/material';

function UserCard({ user }) {
  return (
    <Card sx={{ maxWidth: 345 }}>
      <CardContent>
        <h3>{user.name}</h3>
        <p>{user.email}</p>
        <Button variant="contained" color="primary">
          View Profile
        </Button>
      </CardContent>
    </Card>
  );
}
```

Penjelasan:

- Pre-built components dengan design system yang konsisten
- Mudah digunakan dan customizable
- Mendukung theming

Kesimpulan

- CSS Biasa: Simple, tapi global scope
- CSS Modules: Scoped styling, prevent conflicts
- Styled-Components: CSS-in-JS, highly dynamic
- Tailwind CSS: Utility-first, rapid development
- UI Libraries: Production-ready components

Pilih berdasarkan kebutuhan project dan tim!