

# PRAKTIK PER. 6

## Langkah 1: Setup Project

```
npm create vite@latest praktik-6 -- --template react
```

## Langkah 2: Komponen Lifecycle Demo

File: src/components/LifecycleDemo.jsx



LifecycleDemo.jsx

```
1 import React, { useState, useEffect } from 'react';
2
3 const LifecycleDemo = () => {
4     const [count, setCount] = useState(0);
5     const [isVisible, setIsVisible] = useState(true);
6
7     // Effect tanpa dependencies - run setelah setiap render
8     useEffect(() => {
9         console.log('Effect tanpa dependencies - dijalankan setelah setiap
10    render');
11 });
12
13     // Effect dengan empty dependencies - run sekali setelah mount
14     useEffect(() => {
15         console.log('Component mounted - effect dengan []');
16
17         return () => {
18             console.log('Cleanup dari effect dengan [] - component akan
19 unmount');
20         };
21     }, []);
22
23     // Effect dengan dependency count - run ketika count berubah
24     useEffect(() => {
25         console.log('Count berubah:', count);
26
27         // Update document title
28         document.title = `Count: ${count}`;
29
30         return () => {
31             console.log('Cleanup sebelum count effect dijalankan lagi');
32         };
33     }, [count]);
34
35     // Effect dengan dependency isVisible
36     useEffect(() => {
37         console.log('Visibility berubah:', isVisible);
38     }, [isVisible]);
39
40     const increment = () => {
41         setCount(prev => prev + 1);
42     };
43
44     const toggleVisibility = () => {
45         setIsVisible(prev => !prev);
46     };
47
48     return (
49         <div className="lifecycle-demo">
```

```
48             <h2>Demo Lifecycle dengan useEffect</h2>
49
50             <div className="counter-section">
51                 <h3>Counter: {count}</h3>
52                 <button onClick={increment} className="btn btn-primary">
53                     Tambah Count
54                 </button>
55             </div>
56
57             <div className="visibility-section">
58                 <button onClick={toggleVisibility} className="btn btn-
secondary">
59                     {isVisible ? 'Sembunyikan' : 'Tampilkan'} Pesan
60                 </button>
61
62                 {isVisible && (
63                     <div className="message-box">
64                         <p>Pesan ini {isVisible ? 'terlihat' : 'tersembunyi'}
65                         <p>Count saat ini: {count}</p>
66                     </div>
67                 )}
68             </div>
69
70             <div className="explanation">
71                 <h4>Perhatikan Console Browser:</h4>
72                 <p>Buka Developer Tools (F12) dan lihat console untuk melihat
73                 kapan effects dijalankan</p>
74             </div>
75         </div>
76     );
77 }
78 export default LifecycleDemo;
```

## Langkah 3: Komponen Data Fetching Demo

File: src/components/DataFetchingDemo.jsx

## DataFetchingDemo.jsx

```
1 import React, { useState, useEffect } from 'react';
2
3 const DataFetchingDemo = () => {
4     const [users, setUsers] = useState([]);
5     const [loading, setLoading] = useState(true);
6     const [error, setError] = useState(null);
7     const [userId, setUserId] = useState(1);
8     const [userDetail, setUserDetail] = useState(null);
9
10    // Effect untuk fetching semua users (sekali saat mount)
11    useEffect(() => {
12        const fetchUsers = async () => {
13            try {
14                setLoading(true);
15                setError(null);
16
17                // Simulasi API call dengan timeout
18                await new Promise(resolve => setTimeout(resolve, 1000));
19
20                // Data dummy users
21                const dummyUsers = [
22                    { id: 1, name: 'John Doe', email: 'john@example.com' },
23                    { id: 2, name: 'Jane Smith', email: 'jane@example.com' },
24                    { id: 3, name: 'Bob Johnson', email: 'bob@example.com' },
25                    { id: 4, name: 'Alice Brown', email: 'alice@example.com' }
26                ];
27
28                setUsers(dummyUsers);
29            } catch (err) {
30                setError('Gagal memuat data users');
31                console.error('Error fetching users:', err);
32            } finally {
33                setLoading(false);
34            }
35        };
36
37        fetchUsers();
38    }, []); // Empty dependencies - hanya sekali saat mount
39
40    // Effect untuk fetching user detail berdasarkan userId
41    useEffect(() => {
42        const fetchUserDetail = async () => {
43            if (!userId) return;
44
45            try {
46                setLoading(true);
47
48                // Simulasi API call
49                await new Promise(resolve => setTimeout(resolve, 500));
50            } catch (err) {
51                setError('Gagal memuat detail user');
52            } finally {
53                setLoading(false);
54            }
55        };
56    }, [userId]);
57
58    return (
59        <div>
60            <h1>Data Fetching Demo</h1>
61            <p>User ID:</p>
62            <input type="text" value={userId} onChange={(e) => setUserId(e.target.value)} />
63            <p>User Detail:</p>
64            <pre>{JSON.stringify(userDetail, null, 2)}</pre>
65            <button onClick={() => fetchUserDetail()}>Fetch User Detail</button>
66        </div>
67    );
68}
69
70 export default DataFetchingDemo;
```

```
50
51          // Data dummy user detail
52          const dummyUserDetail = {
53              id: userId,
54              name: `User ${userId}`,
55              email: `user${userId}@example.com`,
56              phone: `+62 812-3456-789${userId}`,
57              address: `Jl. Contoh No. ${userId}, Jakarta`
58          };
59
60          setUserDetail(dummyUserDetail);
61      } catch (err) {
62          setError(`Gagal memuat detail user ${userId}`);
63          console.error('Error fetching user detail:', err);
64      } finally {
65          setLoading(false);
66      }
67  };
68
69  fetchUserDetail();
70 }, [userId]); // Dependency: userId - re-fetch ketika userId berubah
71
72 const handleUserChange = (event) => {
73     setId(parseInt(event.target.value));
74 };
75
76 return (
77     <div className="data-fetching-demo">
78         <h2>Demo Data Fetching dengan useEffect</h2>
79
80         {/* Loading State */}
81         {loading && (
82             <div className="loading">
83                 <div className="spinner"></div>
84                 <p>Memuat data...</p>
85             </div>
86         )}
87
88         {/* Error State */}
89         {error && (
90             <div className="error-message">
91                 {error}
92             </div>
93         )}
94
95         {/* Users List */}
96         <div className="users-section">
97             <h3>Daftar Users</h3>
98             <div className="users-grid">
99                 {users.map(user => (
100                     <div
101                         key={user.id}
102                         className={`user-card ${userId === user.id ?
'active' : ''}}`}
```

```
103             onClick={() => setId(user.id)}
104         >
105             <h4>{user.name}</h4>
106             <p>{user.email}</p>
107         </div>
108     )}
109 </div>
110 </div>
111
112 /* User Detail */
113 <div className="user-detail-section">
114     <h3>Detail User</h3>
115
116     <div className="user-selector">
117         <label>Pilih User ID: </label>
118         <select value={id} onChange={handleUserChange}>
119             {[1, 2, 3, 4].map(id => (
120                 <option key={id} value={id}>
121                     User {id}
122                 </option>
123             )))
124             </select>
125         </div>
126
127         {userDetail && (
128             <div className="user-detail-card">
129                 <h4>{userDetail.name}</h4>
130                 <p><strong>Email:</strong> {userDetail.email}</p>
131                 <p><strong>Phone:</strong> {userDetail.phone}</p>
132                 <p><strong>Address:</strong> {userDetail.address}</p>
133             </div>
134         ))}
135     </div>
136 </div>
137 );
138 };
139
140 export default DataFetchingDemo;
```

## Langkah 4: Komponen Effect Dependencies Demo

File: src/components/DataFetchingDemo.jsx



## EffectDependenciesDemo.jsx

```
1 import React, { useState, useEffect } from 'react';
2
3 const EffectDependenciesDemo = () => {
4     const [count, setCount] = useState(0);
5     const [name, setName] = useState('');
6     const [windowWidth, setWindowWidth] = useState(window.innerWidth);
7     const [timer, setTimer] = useState(0);
8     const [logs, setLogs] = useState([]);
9
10    const addLog = (message) => {
11        setLogs(prev => [...prev, `${new Date().toLocaleTimeString()}: ${message}`]);
12    };
13
14    // Effect 1: Tanpa dependencies - run setelah setiap render
15    useEffect(() => {
16        addLog('Effect tanpa dependencies dijalankan');
17    });
18
19    // Effect 2: Empty dependencies - run sekali setelah mount
20    useEffect(() => {
21        addLog('Effect dengan [] dijalankan (mount)');
22
23        return () => {
24            addLog('Cleanup effect [] (unmount)');
25        };
26    }, []);
27
28    // Effect 3: Dependency count - run ketika count berubah
29    useEffect(() => {
30        addLog(`Effect [count] dijalankan, count: ${count}`);
31
32        return () => {
33            addLog(`Cleanup effect [count] sebelumnya`);
34        };
35    }, [count]);
36
37    // Effect 4: Dependency name - run ketika name berubah
38    useEffect(() => {
39        if (name) {
40            addLog(`Effect [name] dijalankan, name: "${name}"`);
41        }
42    }, [name]);
43
44    // Effect 5: Multiple dependencies - run ketika count atau name berubah
45    useEffect(() => {
46        addLog(`Effect [count, name] dijalankan, count: ${count}, name: ${name}`);
47    }, [count, name]);
```

```
48
49 // Effect 6: Window resize event listener dengan cleanup
50 useEffect(() => {
51     const handleResize = () => {
52         setWindowWidth(window.innerWidth);
53     };
54
55     addLog('Event listener resize dipasang');
56     window.addEventListener('resize', handleResize);
57
58     return () => {
59         addLog('Event listener resize dibersihkan');
60         window.removeEventListener('resize', handleResize);
61     };
62 }, []);
63
64 // Effect 7: Timer dengan cleanup
65 useEffect(() => {
66     addLog('Timer dimulai');
67
68     const interval = setInterval(() => {
69         setTimer(prev => prev + 1);
70     }, 1000);
71
72     return () => {
73         addLog('Timer dihentikan');
74         clearInterval(interval);
75     };
76 }, []);
77
78 const resetLogs = () => {
79     setLogs([]);
80 };
81
82 return (
83     <div className="dependencies-demo">
84         <h2>Demo Effect Dependencies</h2>
85
86         <div className="controls">
87             <div className="control-group">
88                 <label>Count: {count}</label>
89                 <button onClick={() => setCount(prev => prev + 1)}>
90                     +1 Count
91                 </button>
92             </div>
93
94             <div className="control-group">
95                 <label>Name:</label>
96                 <input
97                     type="text"
98                     value={name}
99                     onChange={(e) => setName(e.target.value)}
100                    placeholder="Ketik nama...">

```

```
101             className="text-input"
102         />
103     </div>
104
105     <div className="control-group">
106         <label>Window Width: {windowWidth}px</label>
107     </div>
108
109     <div className="control-group">
110         <label>Timer: {timer}s</label>
111     </div>
112
113     <button onClick={resetLogs} className="btn btn-secondary">
114         Reset Logs
115     </button>
116 </div>
117
118     <div className="explanation">
119         <h3>Penjelasan Dependencies:</h3>
120         <ul>
121             <li><strong>No dependencies</strong> - Run setelah setiap
122             render</li>
123             <li><strong>[]</strong> - Run sekali setelah mount</li>
124             <li><strong>[count]</strong> - Run ketika count
125             berubah</li>
126             <li><strong>[name]</strong> - Run ketika name berubah</li>
127             <li><strong>[count, name]</strong> - Run ketika count ATAU
128             name berubah</li>
129         </ul>
130     </div>
131
132     <div className="logs-container">
133         <h3>Effect Execution Logs:</h3>
134         <div className="logs">
135             {logs.slice(-10).map((log, index) => (
136                 <div key={index} className="log-entry">
137                     {log}
138                 </div>
139             )));
140             {logs.length === 0 && <p>Belum ada logs...</p>}
141         </div>
142         <p className="log-count">Total logs: {logs.length}</p>
143     </div>
144   );
145 };
```

145 export default EffectDependenciesDemo;

## Langkah 5: Styling CSS

File: src/App.css

## EffectDependenciesDemo.jsx

```
1 /* File: src/App.css */
2 .App {
3   max-width: 1200px;
4   margin: 0 auto;
5   padding: 20px;
6   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
7 }
8
9 /* Lifecycle Demo Styles */
10 .lifecycle-demo {
11   background: #f8f9fa;
12   padding: 20px;
13   border-radius: 10px;
14   margin-bottom: 30px;
15   border-left: 4px solid #007bff;
16 }
17
18 .counter-section,
19 .visibility-section {
20   margin: 20px 0;
21   padding: 15px;
22   background: white;
23   border-radius: 8px;
24   border: 1px solid #dee2e6;
25 }
26
27 .message-box {
28   background: #e7f3ff;
29   padding: 15px;
30   border-radius: 5px;
31   margin-top: 10px;
32   border-left: 4px solid #007bff;
33 }
34
35 .explanation {
36   background: #fff3cd;
37   padding: 15px;
38   border-radius: 5px;
39   border: 1px solid #ffeaaf;
40 }
41
42 /* Data Fetching Demo Styles */
43 .data-fetching-demo {
44   background: #f8f9fa;
45   padding: 20px;
46   border-radius: 10px;
47   margin-bottom: 30px;
48   border-left: 4px solid #28a745;
49 }
```

```
50
51 .loading {
52   display: flex;
53   align-items: center;
54   gap: 10px;
55   background: #e7f3ff;
56   padding: 15px;
57   border-radius: 5px;
58   margin: 10px 0;
59 }
60
61 .spinner {
62   width: 20px;
63   height: 20px;
64   border: 2px solid #007bff;
65   border-top: 2px solid transparent;
66   border-radius: 50%;
67   animation: spin 1s linear infinite;
68 }
69
70 @keyframes spin {
71   0% {
72     transform: rotate(0deg);
73   }
74
75   100% {
76     transform: rotate(360deg);
77   }
78 }
79
80 .error-message {
81   background: #f8d7da;
82   color: #721c24;
83   padding: 15px;
84   border-radius: 5px;
85   margin: 10px 0;
86   border: 1px solid #f5c6cb;
87 }
88
89 .users-grid {
90   display: grid;
91   grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
92   gap: 10px;
93   margin: 15px 0;
94 }
95
96 .user-card {
97   background: white;
98   padding: 15px;
99   border-radius: 8px;
100  border: 2px solid #dee2e6;
101  cursor: pointer;
102  transition: all 0.3s ease;
103 }
```

```
104
105 .user-card:hover {
106   border-color: #007bff;
107   transform: translateY(-2px);
108 }
109
110 .user-card.active {
111   border-color: #28a745;
112   background: #f8ffff;
113 }
114
115 .user-detail-section {
116   margin-top: 20px;
117   padding: 20px;
118   background: white;
119   border-radius: 8px;
120   border: 1px solid #dee2e6;
121 }
122
123 .user-selector {
124   margin-bottom: 15px;
125 }
126
127 .user-detail-card {
128   background: #f8f9fa;
129   padding: 15px;
130   border-radius: 5px;
131   border-left: 4px solid #28a745;
132 }
133
134 /* Dependencies Demo Styles */
135 .dependencies-demo {
136   background: #f8f9fa;
137   padding: 20px;
138   border-radius: 10px;
139   border-left: 4px solid #ffc107;
140 }
141
142 .controls {
143   display: grid;
144   grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
145   gap: 15px;
146   margin: 20px 0;
147   padding: 20px;
148   background: white;
149   border-radius: 8px;
150   border: 1px solid #dee2e6;
151 }
152
153 .control-group {
154   display: flex;
155   flex-direction: column;
156   gap: 5px;
157 }
```

```
158
159 .text-input {
160   padding: 8px 12px;
161   border: 2px solid #e9ecef;
162   border-radius: 5px;
163   font-size: 14px;
164 }
165
166 .text-input:focus {
167   outline: none;
168   border-color: #007bff;
169 }
170
171 .logs-container {
172   margin-top: 20px;
173 }
174
175 .logs {
176   max-height: 300px;
177   overflow-y: auto;
178   background: #2d3748;
179   color: #e2e8f0;
180   padding: 15px;
181   border-radius: 5px;
182   font-family: 'Courier New', monospace;
183   font-size: 14px;
184 }
185
186 .log-entry {
187   padding: 5px 0;
188   border-bottom: 1px solid #4a5568;
189 }
190
191 .log-entry:last-child {
192   border-bottom: none;
193 }
194
195 .log-count {
196   text-align: center;
197   margin-top: 10px;
198   font-style: italic;
199   color: #6c757d;
200 }
201
202 /* Common Button Styles */
203 .btn {
204   padding: 10px 15px;
205   border: none;
206   border-radius: 5px;
207   cursor: pointer;
208   font-weight: bold;
209   transition: all 0.3s ease;
210 }
211
```

```
212 .btn-primary {  
213   background: #007bff;  
214   color: white;  
215 }  
216  
217 .btn-secondary {  
218   background: #6c757d;  
219   color: white;  
220 }  
221  
222 .btn-success {  
223   background: #28a745;  
224   color: white;  
225 }  
226  
227 .btn:hover {  
228   opacity: 0.9;  
229   transform: translateY(-1px);  
230 }  
231  
232 /* Responsive Design */  
233 @media (max-width: 768px) {  
234   .controls {  
235     grid-template-columns: 1fr;  
236   }  
237  
238   .users-grid {  
239     grid-template-columns: 1fr;  
240   }  
241 }
```

## Langkah 5: App.jsx

```
App.jsx
```

```
1 import React from 'react';
2 import './App.css';
3 import LifecycleDemo from './components/LifecycleDemo';
4 import DataFetchingDemo from './components/DataFetchingDemo';
5 import EffectDependenciesDemo from './components/EffectDependenciesDemo';
6
7 function App() {
8   return (
9     <div className="App">
10      <h1>Praktik useEffect dan Lifecycle - Pertemuan 6</h1>
11      <LifecycleDemo />
12      <DataFetchingDemo />
13      <EffectDependenciesDemo />
14    </div>
15  );
16}
17
18 export default App;
```

## Tampilan

### Praktik useEffect dan Lifecycle - Pertemuan 6

#### Demo Lifecycle dengan useEffect

Counter: 0

Tambah Count

Sembunyikan Pesan

Pesan ini terlihat

Count saat ini: 0

Perhatikan Console Browser:

Buka Developer Tools (F12) dan lihat console untuk melihat kapan effects dijalankan

## Demo Data Fetching dengan useEffect

### Daftar Users

John Doe john@example.com	Jane Smith jane@example.com	Bob Johnson bob@example.com	Alice Brown alice@example.com
------------------------------	--------------------------------	--------------------------------	----------------------------------

### Detail User

Pilih User ID:

#### User 1

Email: user1@example.com

Phone: +62 812-3456-7891

Address: Jl. Contoh No. 1, Jakarta

## Demo Effect Dependencies

Count: 0

Name:

Window Width: 1600px

Timer: 39s

+1 Count

Ketik nama...

Reset Logs

### Penjelasan Dependencies:

- **No dependencies** - Run setelah setiap render
- **[]** - Run sekali setelah mount
- **[count]** - Run ketika count berubah
- **[name]** - Run ketika name berubah
- **[count, name]** - Run ketika count ATAU name berubah

### Effect Execution Logs:

```
5:09:42 PM: Effect tanpa dependencies dijalankan
```

Total logs: 51474