

# React Development Professional

Prüfungscode IFW-REACT-300-01 · Fassung 1.0 · Stand 18.09.2026

React für anspruchsvolle Anwendungen: Effekte und ihr Lebenszyklus, State und Formulare, Rendering und Reconciliation sowie Performance und Code-Splitting.

## Die Prüfung im Überblick

|                                                   |                                                   |
|---------------------------------------------------|---------------------------------------------------|
| Bearbeitungszeit<br><b>2 Stunden</b>              | Fragen<br><b>70</b>                               |
| Bestehensgrenze<br><b>70 %</b>                    | Fragetypen<br><b>Einfach- und Mehrfachauswahl</b> |
| Prüfungssprache<br><b>Deutsch</b>                 | Zertifikat gültig<br><b>36 Monate</b>             |
| Durchführung<br><b>Online, zeitlich überwacht</b> | Preis<br><b>99,00 €</b>                           |

## Themenblöcke

| Block         | Thema                          | Gewichtung   | Fragen    |
|---------------|--------------------------------|--------------|-----------|
| <b>1</b>      | Effekte und Lebenszyklus       | 40 %         | 28        |
| <b>2</b>      | State und Formulare            | 20 %         | 14        |
| <b>3</b>      | Rendering und Reconciliation   | 20 %         | 14        |
| <b>4</b>      | Performance und Code-Splitting | 20 %         | 14        |
| <b>Gesamt</b> |                                | <b>100 %</b> | <b>70</b> |

Die Gewichtung gibt an, welchen Anteil ein Block an den Prüfungsfragen hat. Die Fragen werden beim Start zufällig und anteilig je Block aus dem Fragenpool gezogen.

## Effekte und Lebenszyklus

*Effekte korrekt synchronisieren, aufräumen und gegen veraltete Werte absichern.*

- 1.1** Effekte aufräumen  
Cleanup-Funktion bei geänderten Abhängigkeiten und beim Entfernen der Komponente
- 1.2** Das Abhängigkeitsarray richtig füllen  
fehlende Abhängigkeiten, leeres Array, Vergleich mit `Object.is`
- 1.3** Veraltete Werte vermeiden  
Stale Closures bei `setInterval`, funktionale Updates
- 1.4** Race Conditions beim Datenladen verhindern  
Ignore-Flag, `AbortController`; keine `async`-Funktion als Effekt
- 1.5** `useLayoutEffect` gezielt einsetzen  
läuft synchron vor dem Zeichnen, etwa zum Messen des Layouts
- 1.6** Das Verhalten im `StrictMode` verstehen  
Effekte laufen in der Entwicklung zweimal an

## State und Formulare

*State so modellieren, dass Aktualisierungen vorhersagbar bleiben.*

- 2.1** State-Updates bündeln  
Batching, funktionale Updates wie `setCount(c => c + 1)`
- 2.2** State mit Reducern verwalten  
`useReducer`, unveränderliche Updates
- 2.3** Refs und State unterscheiden  
`useRef` löst kein neues Rendern aus
- 2.4** Formulare umsetzen  
kontrollierte und unkontrollierte Elemente, `value` und `defaultValue`

## Rendering und Reconciliation

*Wie React entscheidet, welche Komponenten erhalten, aktualisiert oder neu erzeugt werden.*

- 3.1** Reconciliation nachvollziehen  
Vergleich nach Typ und Position im Baum
- 3.2** Keys richtig wählen  
stabile Keys statt Index, State über `key` zurücksetzen

### 3.3 Fehler in Komponenten abfangen

Error Boundaries und was sie nicht abfangen

---

## BLOCK 4

20 % · 14 Fragen

## Performance und Code-Splitting

*Unnötige Arbeit vermeiden und große Anwendungen in Teilen laden.*

- 4.1 Memoisierung einsetzen  
React.memo, useMemo, useCallback und Referenzgleichheit
- 4.2 Context gezielt einsetzen  
alle Konsumenten rendern neu, wenn sich der Wert ändert
- 4.3 Updates priorisieren  
useTransition, startTransition
- 4.4 Code nachladen  
React.lazy und Suspense

*Institut für Webentwicklung · unabhängiges Lernangebot ohne Verbindung zu den genannten Technologieanbietern. Maßgeblich ist die Fassung auf der Zertifizierungsseite.*