Unmounting

Phase de Unmounting

Le composant est retiré du DOM. C'est le moment de nettoyer les effets de bord : timers, abonnements, listeners...

Avec les classes

```
class Timer extends Component {
  componentDidMount() {
    this.interval = setInterval(() => {
      this.setState(prev => ({ count: prev.count + 1 }));
    }, 1000);
  }

  componentWillUnmount() {
    // Nettoyage obligatoire pour éviter les memory leaks
    clearInterval(this.interval);
  }

  render() { return <p>{this.state.count}</p>; }
}
```

Avec les Hooks (fonction de cleanup)

```
function Timer() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setCount(prev => prev + 1);
    }, 1000);
```

```
    // La fonction retournée = cleanup (unmount OU avant le prochain effet)
    return () => clearInterval(interval);
  }, []);

  return <p>{count}</p>;
}
```

Cas d'usage typiques du cleanup

- Annuler des requêtes HTTP (`AbortController`)
- Désabonner d'un WebSocket / EventEmitter
- Retirer des event listeners (`removeEventListener`)
- Effacer des timers (`clearTimeout`, `clearInterval`)

```
useEffect(() => {
  const controller = new AbortController();

  fetch('/api/data', { signal: controller.signal })
    .then(res => res.json())
    .then(setData)
    .catch(err => {
      if (err.name !== 'AbortError') console.error(err);
    });

  return () => controller.abort(); // Annule la requête si démontage
}, []);
```

Revision #1
Created 29 March 2026 12:57:34 by Hugo
Updated 29 March 2026 12:57:34 by Hugo