Custom Hooks

Custom Hooks

Un custom hook est une fonction commençant par `use` qui encapsule de la logique réutilisable avec des hooks React.

useLocalStorage

```
function useLocalStorage(key, initialValue) {
  const [value, setValue] = useState(() => {
    try {
      const stored = localStorage.getItem(key);
      return stored ? JSON.parse(stored) : initialValue;
    } catch { return initialValue; }
  });

  const setStoredValue = (newValue) => {
    setValue(newValue);
    localStorage.setItem(key, JSON.stringify(newValue));
  };

  return [value, setStoredValue];
}

// Usage
const [theme, setTheme] = useLocalStorage('theme', 'light');
```

useFetch

```
function useFetch(url) {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
```

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

    fetch(url, { signal: controller.signal })
      .then(res => { if (!res.ok) throw new Error(res.statusText); return res.json(); })
      .then(data => { setData(data); setLoading(false); })
      .catch(err => { if (err.name !== 'AbortError') { setError(err); setLoading(false); } });

    return () => controller.abort();
  }, [url]);

  return { data, loading, error };
}

// Usage
const { data: users, loading } = useFetch('/api/users');
```

useDebounce

```
function useDebounce(value, delay = 500) {
  const [debouncedValue, setDebouncedValue] = useState(value);

  useEffect(() => {
    const timer = setTimeout(() => setDebouncedValue(value), delay);
    return () => clearTimeout(timer);
  }, [value, delay]);

  return debouncedValue;
}

// Usage : recherche sans spammer l'API
function Recherche() {
  const [query, setQuery] = useState('');
  const debouncedQuery = useDebounce(query, 300);

  const { data } = useFetch(`/api/search?q=${debouncedQuery}`);
}
```

Revision #1
Created 29 March 2026 12:57:40 by Hugo
Updated 29 March 2026 12:57:40 by Hugo