Custom Hooks 
Functions that contain some reusable logic 
Custom hooks usually reuse built-in hooks (like 
useState, useEffect) 
Usually easiest to extract logic into a hook, 
rather than making a hook first 

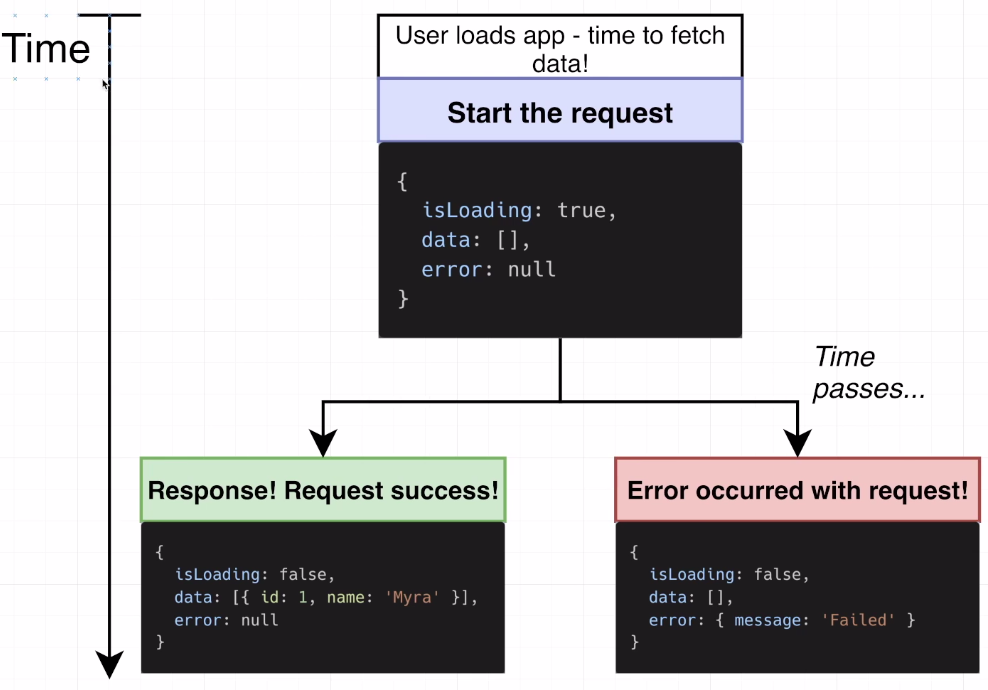
CounterPage 
count 
useEffect 
handleClick 
JSX 
I 
Three chunks of code that... 
- Creates number state based on 
an initial value 
Logs that value any time it 
changes 
- Provides a way to change that 
value 
Seems kind of useful! 
Maybe we should create 
a hook! 

CounterPage 
useCounter() 
JSX 
useCounter 
count 
useEffect 
handleClick 

1 
2 
3 
4 
5 
6 
7 
Make a function called 'useSomething' 
Find all the non-JSX expressions that refer to 1-2 
related pieces of state 
Cut them all out, paste them into 'useSomething' 
Find 'not defined' errors in your component. 
In your hook, return an object that contains the 
variables the component needs 
In the component, call your hook. Destructure the 
properties the component needs 
Find 'not defined' errors in the hook. Pass the missing 
variables in as arguments to the hook 

Thunks

It is a function that delays the execution of a function or block of code until it is called again. In Redux, it is used to manage async operations (e.g. fetching data or database operations). With this extension, managing async operations become easier and your application can become more flexible.



Redux Store 
isLoading: true, 
Time 
Start the 
request 
Error with 
request! 
Automatically 
dispatches 
actions during 
data loading 
Async 
T hunk 
Function 
Async 
Thunk 
Function 
Action 
type 
pending 
'Pending' means we 
are in the process of 
fetching data 
Action 
type rejected 
Rejected' means an 
error occurred during 
the request 
Dispatch 
Dispatch 
data: [l, 
Reducers 
error: null 
isLoading: false, 
Reducers 
data: Il, 
error: message: 
Redux Store 
'Fatted' 

Redux Store 
Time 
Start the 
request 
Response! 
Request 
success! 
Automatically 
dispatches 
actions during 
data loading 
Async 
Thunk 
Function 
Async 
Thunk 
Function 
Action 
pending 
type 
'Pending' means we 
are in the process of 
fetching data 
Action 
type fulfilled 
fulfilled' means we 
fetched data 
successfully 
Dispatch 
Dispatch 
Reducers 
Reducers 
isLoading: true, 
data: [l, 
error: nutt 
isLoading.• false, 
data: id: I, name: 
error: null 
'Myra' Il, 
Redux Store 

