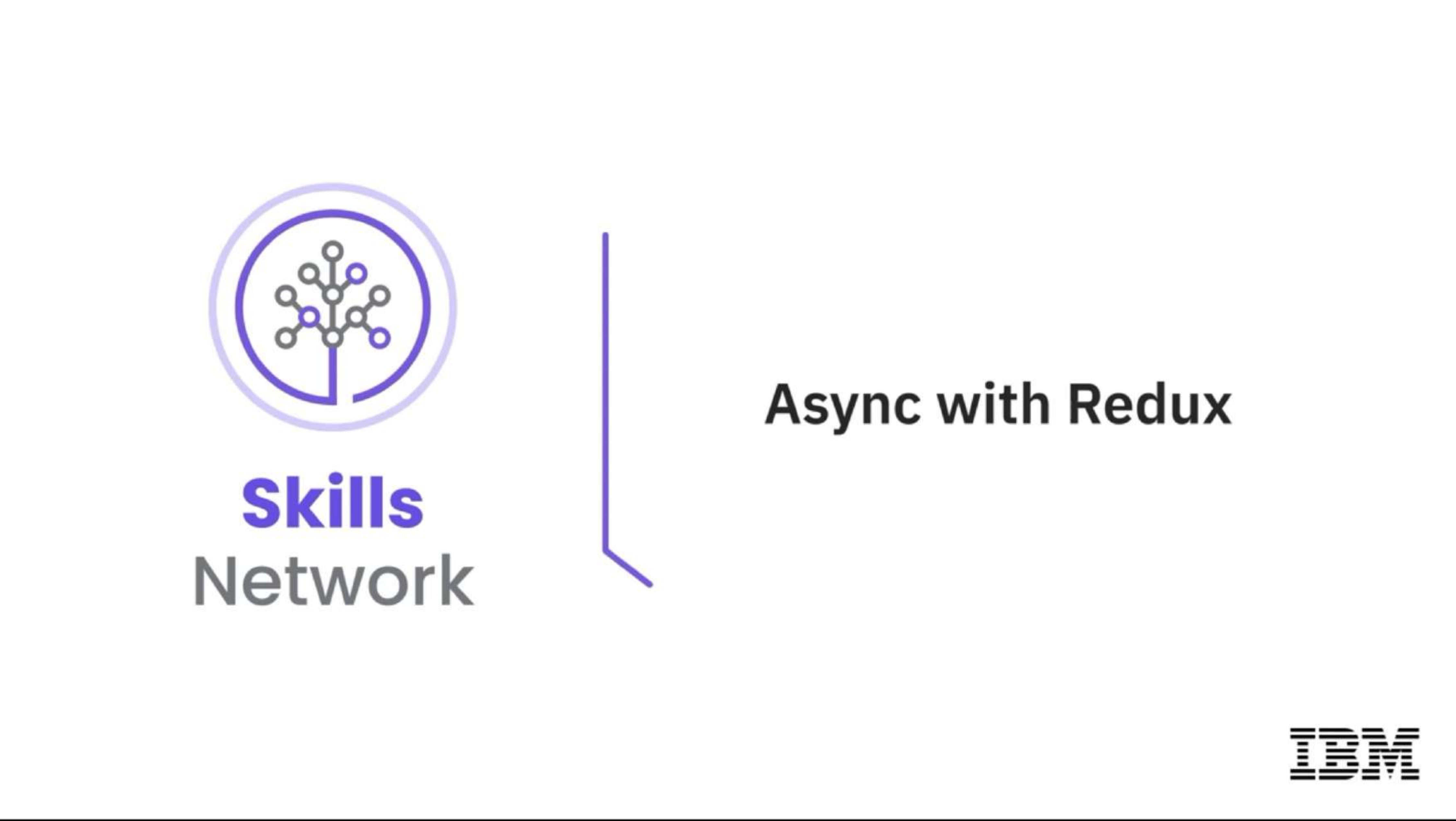


Skills Network

Async with Redux

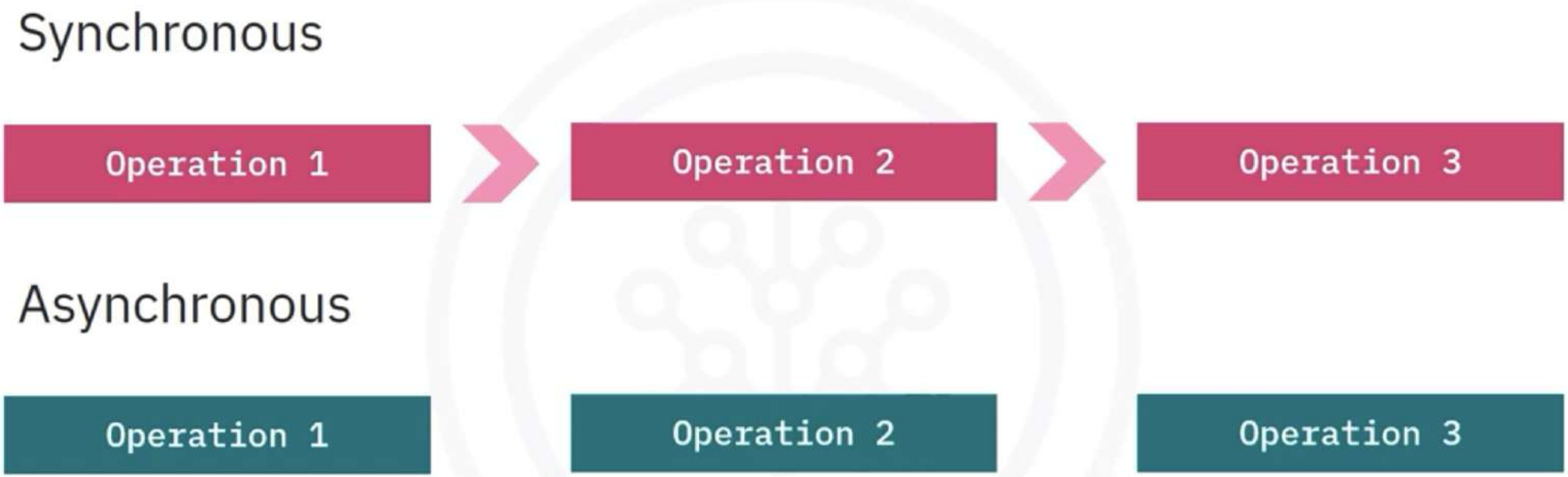
IBM

Async with Redux

What you will learn

- Explain the differences between sync and async
- Explain challenges with async operations in Redux
- Provide examples of complications posed by async operations
- Describe how you can use Thunk and Saga to handle async
- Compare and contrast Thunk and Saga's pros and cons

Operation behavior



Synchronous

Operation 1

Operation 2

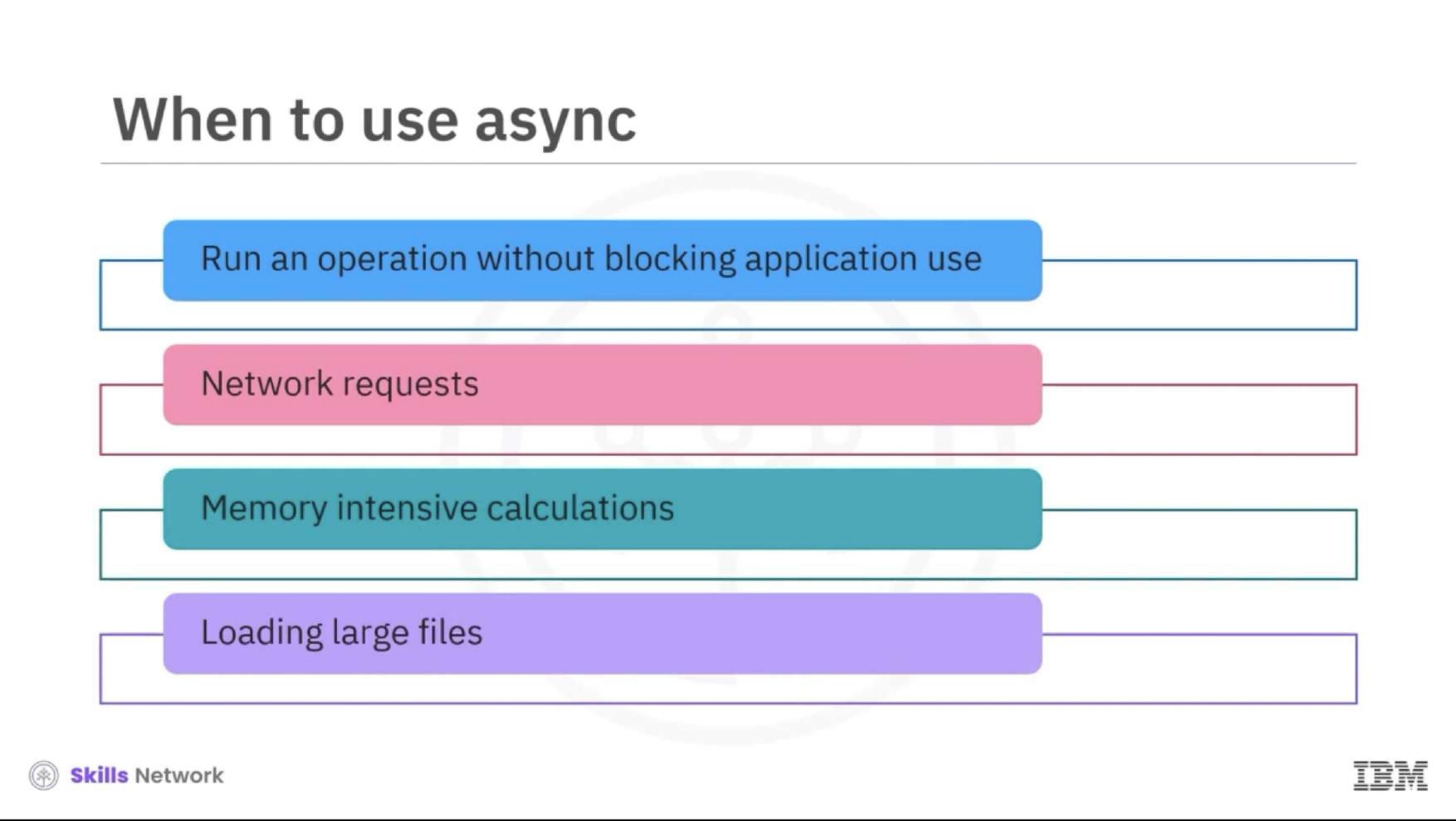
Operation 3

Asynchronous

Operation 1

Operation 2

Operation 3

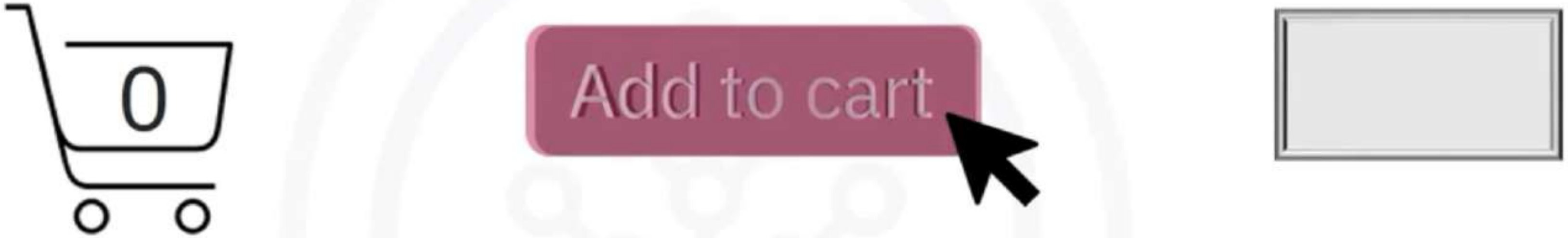


When to use async

- Run an operation without blocking application use
- Network requests
- Memory intensive calculations
- Loading large files

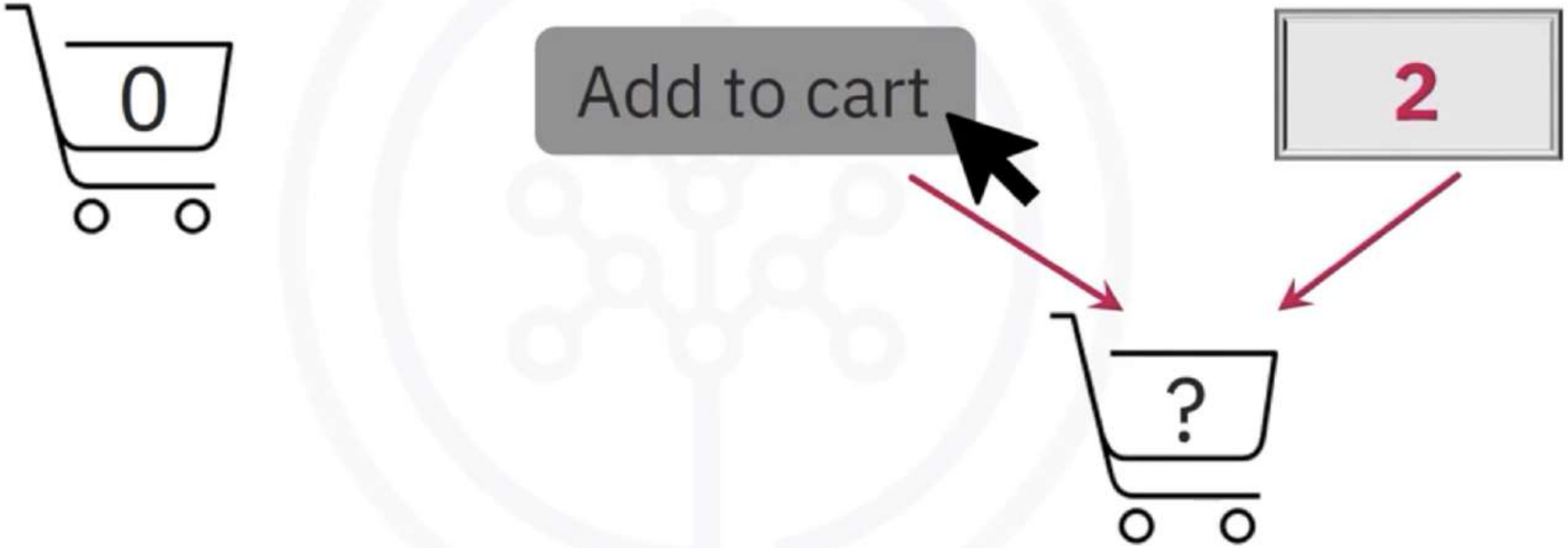
Problems with async example

Asynchronous behavior



Problems with async example

Asynchronous behavior



Redux behavior

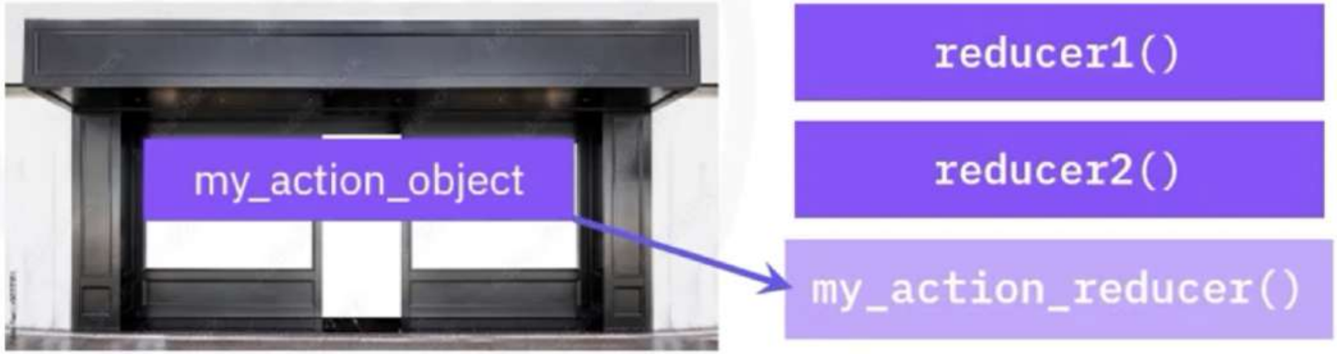
Redux requires sync

```
my_action_creator()
  return{

      my_action_object

  }
```

Redux behavior

Redux requires sync

```
my_action_creator()
  return{


  }
```



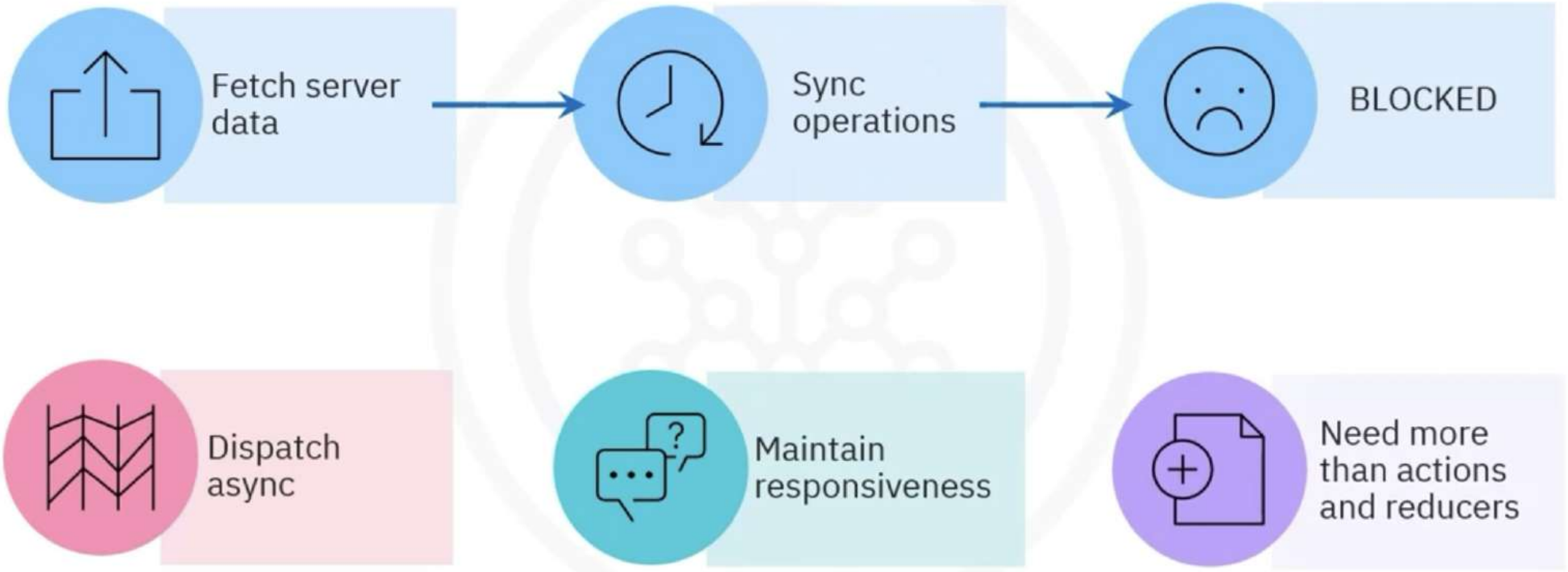
my_action_object

reducer1()

reducer2()

my_action_reducer()

Async dispatching



Fetch server data → Sync operations → BLOCKED

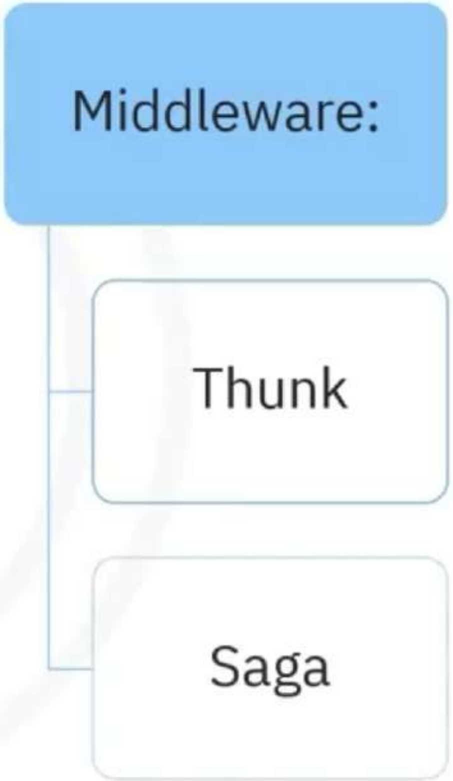
Dispatch async

Maintain responsiveness

Need more than actions and reducers

Middleware

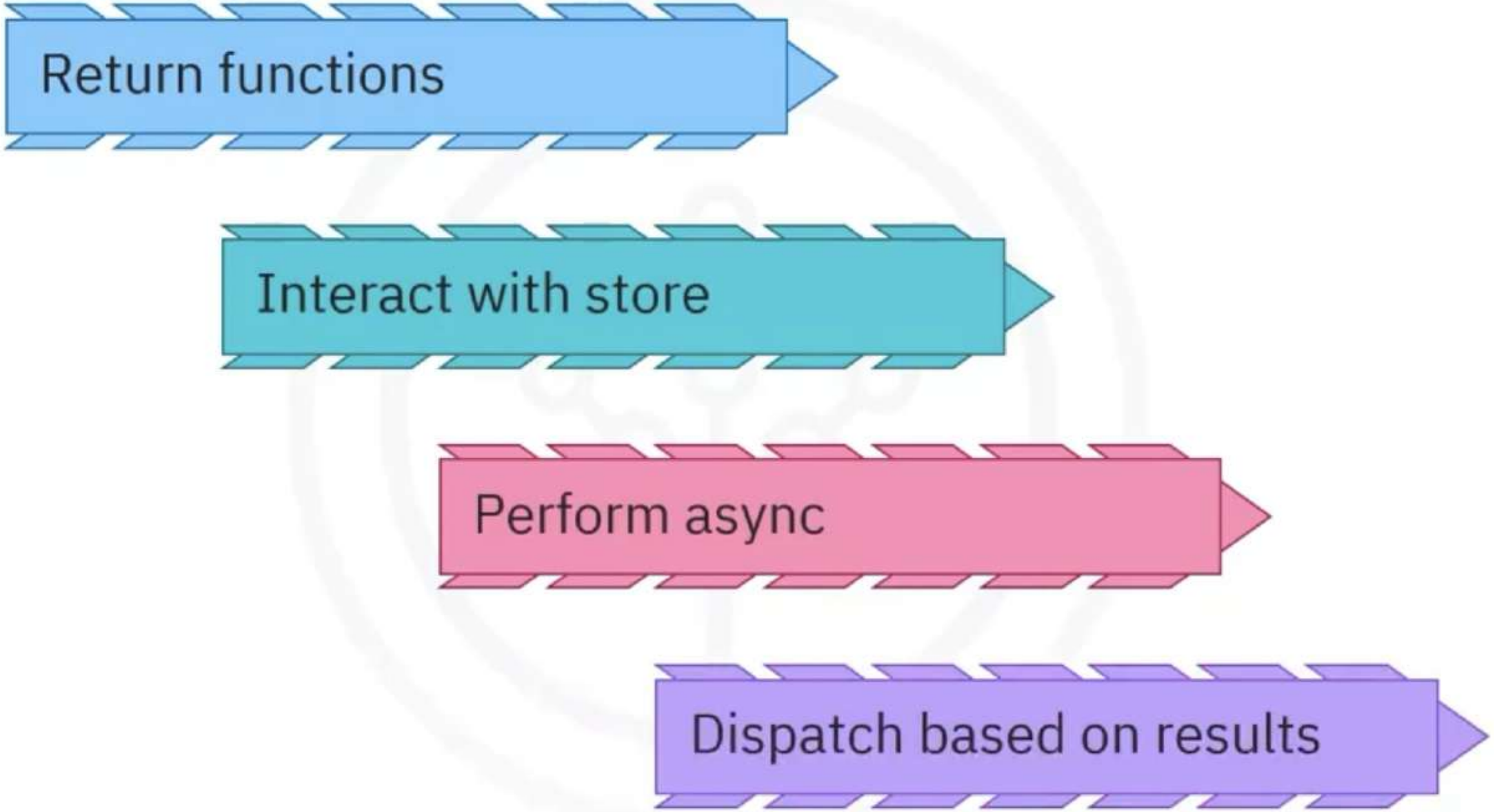
- Middleware intercepts actions
- Delays if necessary
- Continue when async completes
- The last action restores the sync flow


Middleware:

Thunk

Saga

Thunk

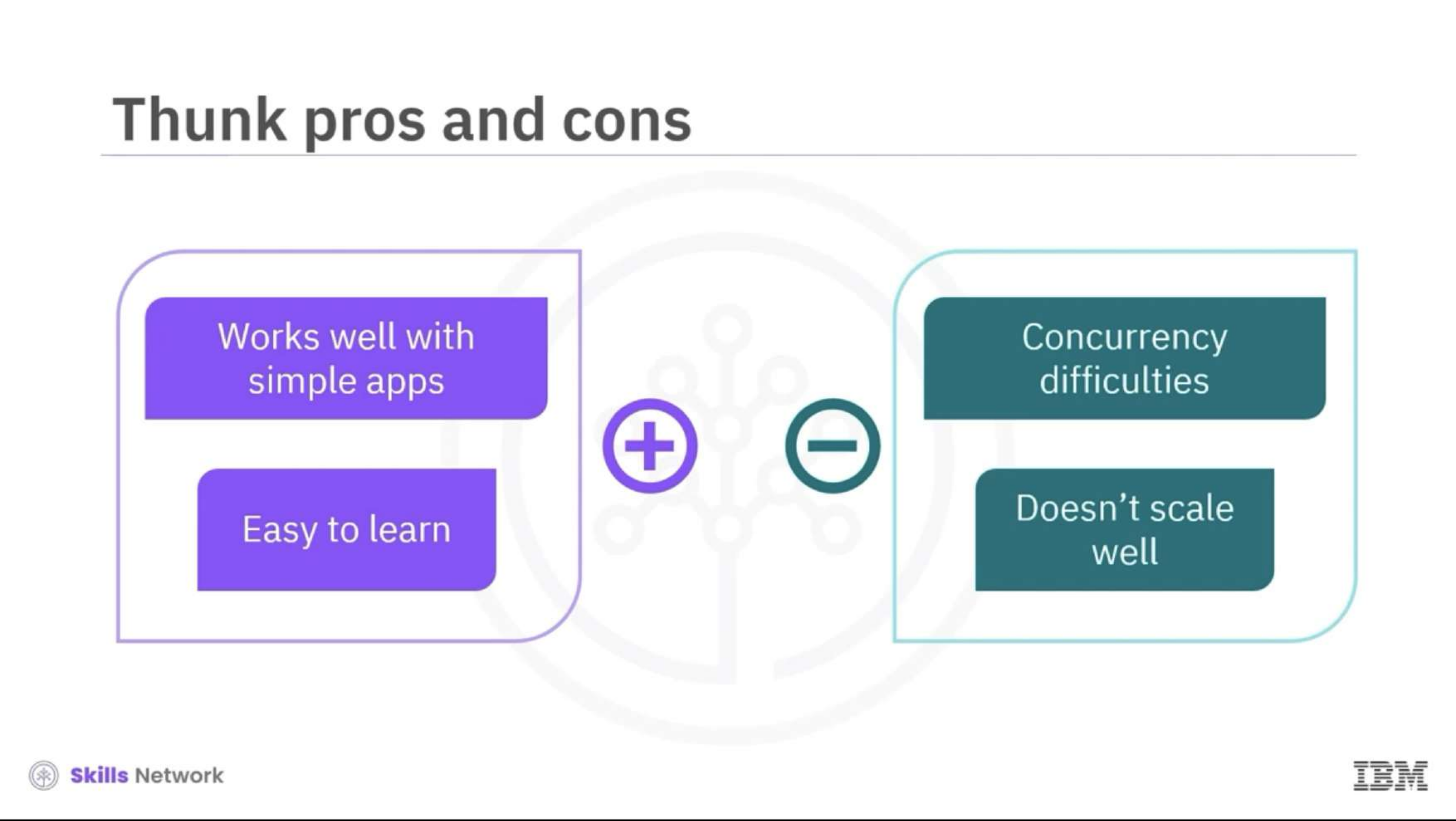


Return functions

Interact with store

Perform async

Dispatch based on results



Thunk pros and cons

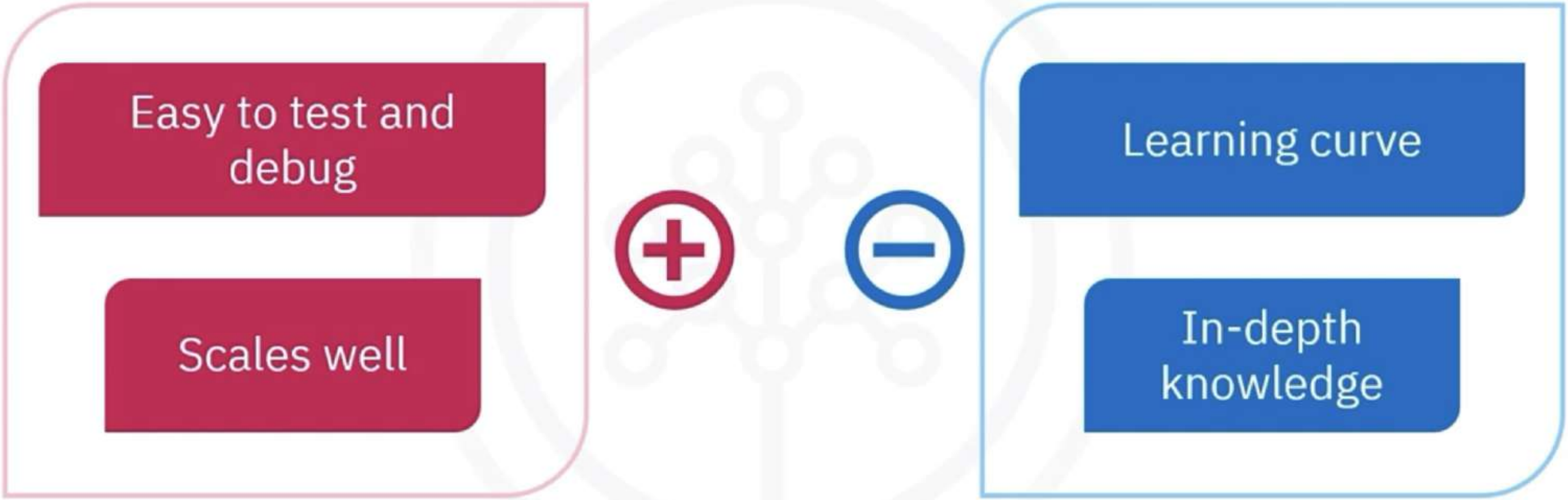
- Works well with simple apps
- Easy to learn

- Concurrency difficulties
- Doesn't scale well

Saga

- Generator functions
- Called Sagas
- Encapsulate async logic
- Multitask with dispatch process
- Pause and resume as needed

Saga pros and cons

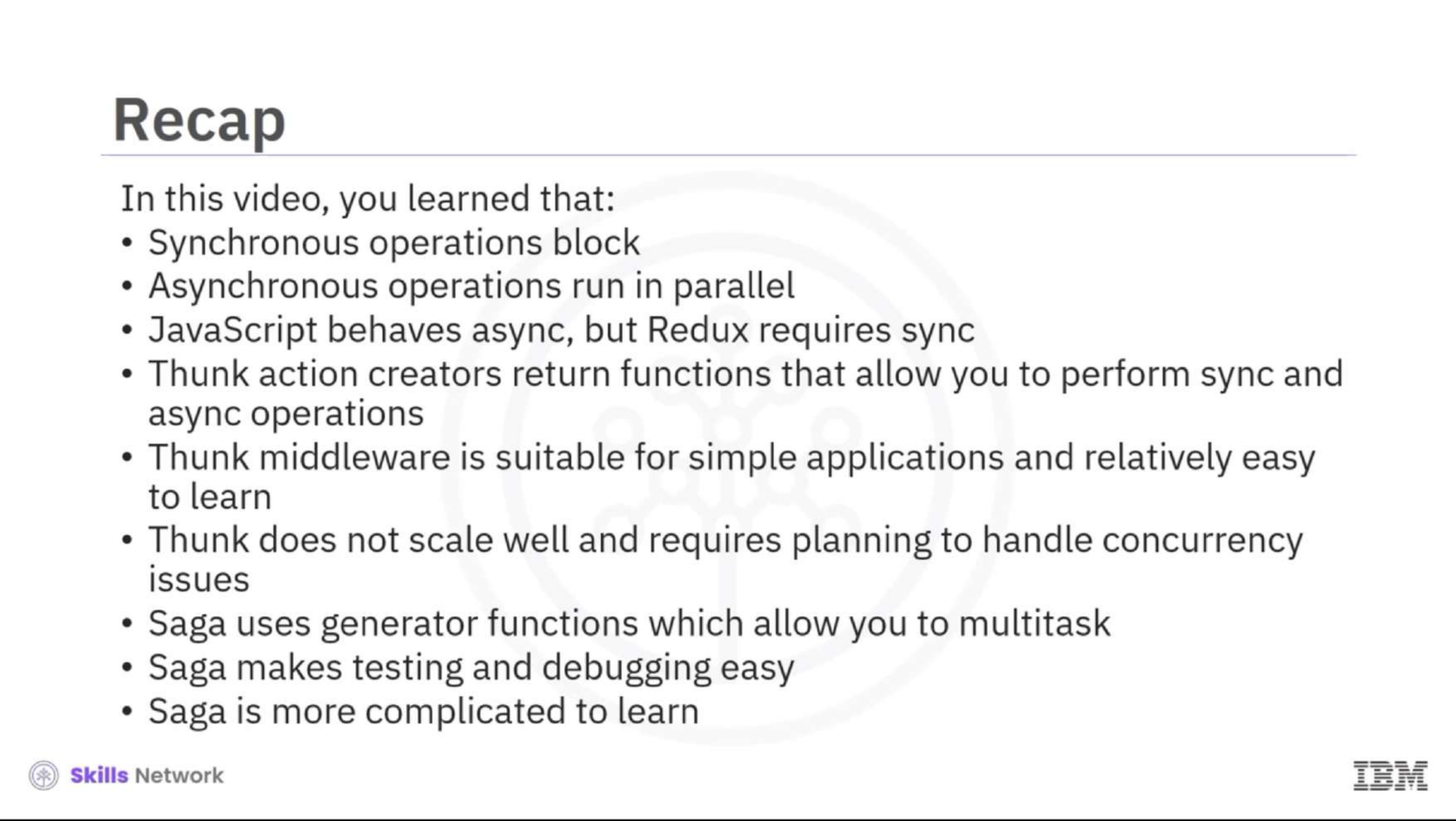


Easy to test and debug

Scales well

Learning curve

In-depth knowledge



Recap

In this video, you learned that:

- Synchronous operations block
- Asynchronous operations run in parallel
- JavaScript behaves async, but Redux requires sync
- Thunk action creators return functions that allow you to perform sync and async operations
- Thunk middleware is suitable for simple applications and relatively easy to learn
- Thunk does not scale well and requires planning to handle concurrency issues
- Saga uses generator functions which allow you to multitask
- Saga makes testing and debugging easy
- Saga is more complicated to learn