Skills Network

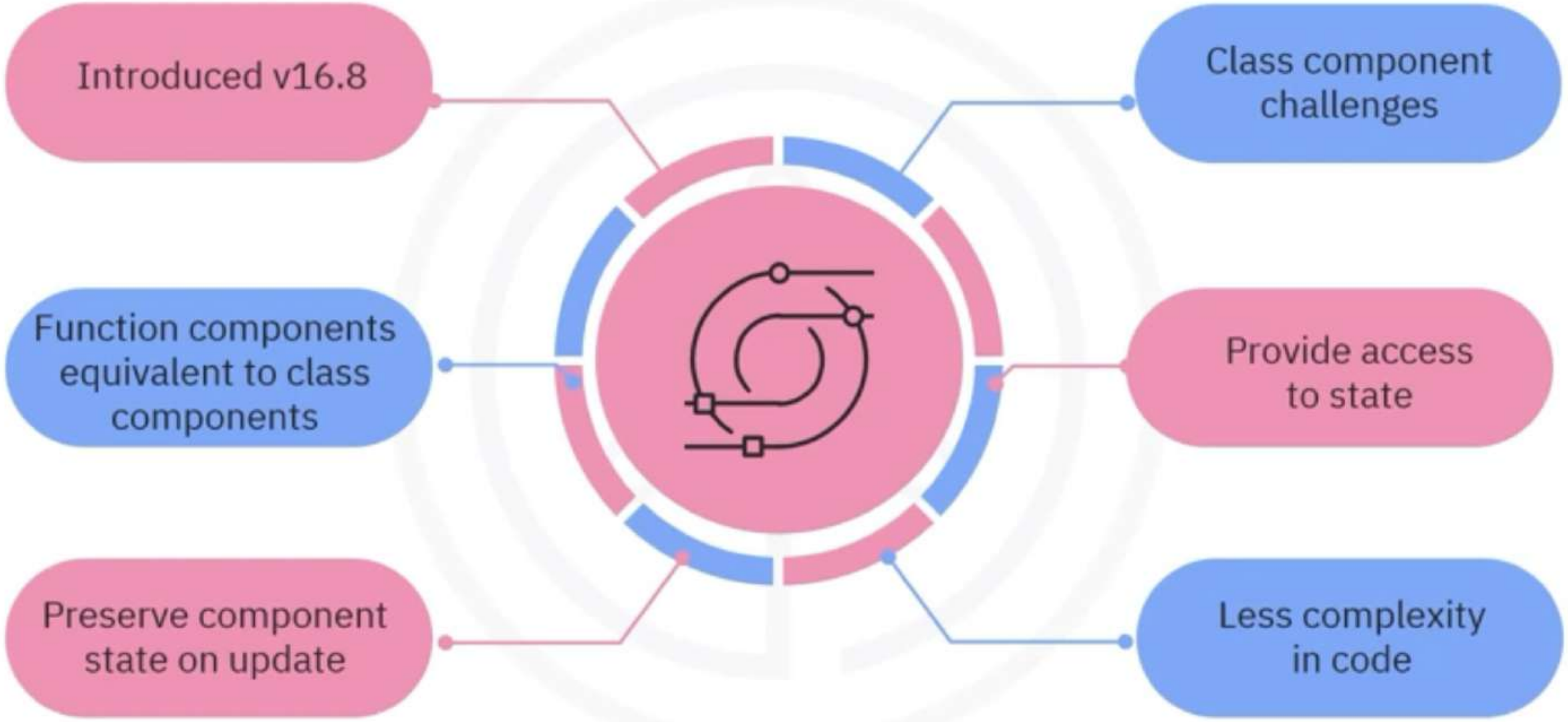
Hooks

IBM

What you will learn

- Explain the purpose of using hooks
- List the advantages of hooks
- Summarize best practices for hook development
- Contrast standard and custom hooks

Introduction and purpose


Introduced v16.8
Function components equivalent to class components
Preserve component state on update
Class component challenges
Provide access to state
Less complexity in code

Custom hooks

Write your own hooks

Prevents code duplication

Use across multiple components

Advantages

- Readable
- Less code
- Optimized
- Simplifies code
- Handles events and logic
- Performance boost

Best practices

Hooks

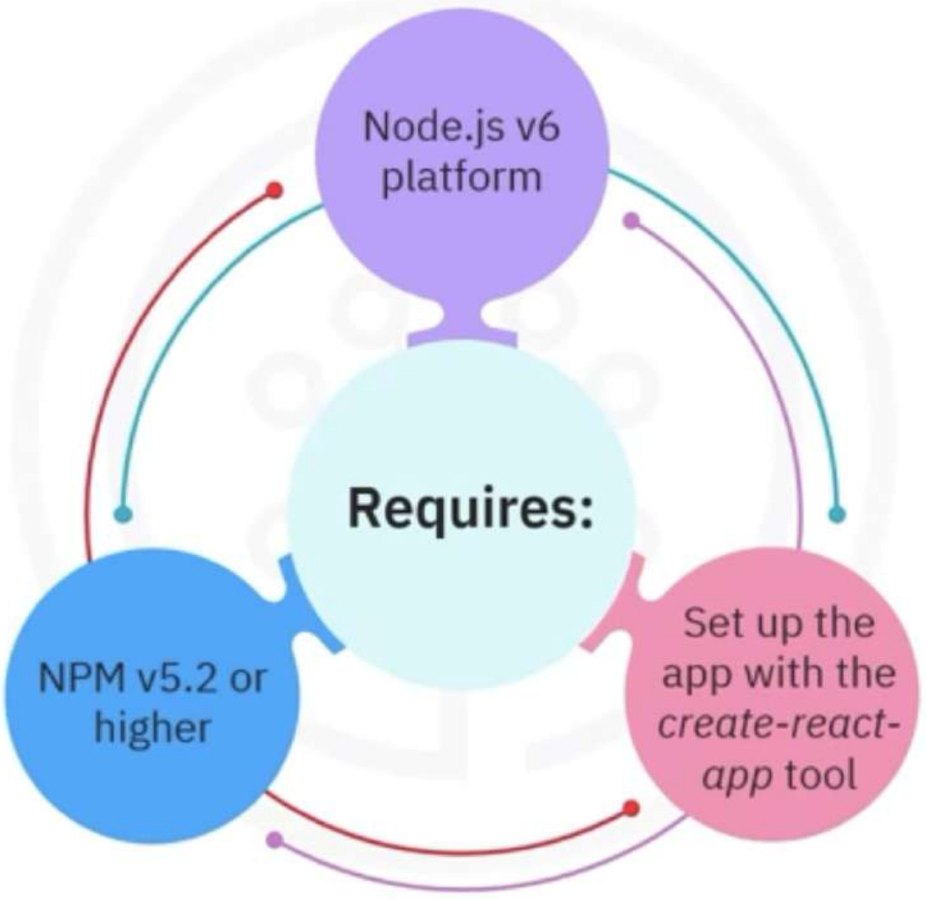
Do:

- Use only with function components
- Call only at top of component tree

Do not:

- Use with normal JS functions
- Use inside loops, conditionals, or nested functions

For use with


Node.js v6 platform
Requires:
NPM v5.2 or higher
Set up the app with the *create-react-app* tool

Common hooks

useState: Adds state to a function component

useEffect: Manages side effects

useContext: Manages context changes

useReducer: Manages Redux state changes

Writing custom hooks

Prefix with `'use'`
Ex:
`useLocalStorage`
`useAuthentication`

Same features as normal JS functions

Composed with one or more hooks

Reuse and combine

Example of hooks

```
import React, { useState } from 'react';
function CntApp() {
  // Declare a new state variable "count"
  const [count, setCount] = useState(0);
  return (
    <div>
      <p>You clicked {count} many times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
export default CntApp;
```

Recap

In this video, you learned that:

- Hooks provide function components with the same capabilities as class components
- Hooks enable you to write simpler, more readable, and a lesser amount of code, providing more complex behaviors and improving performance
- You must call hooks at the top of a tree and cannot call them from regular or nested functions or inside loops or conditions
- Standard hooks include useState, useEffect, useContext, and useReducer
- And finally, you can add unique functionality using custom hooks