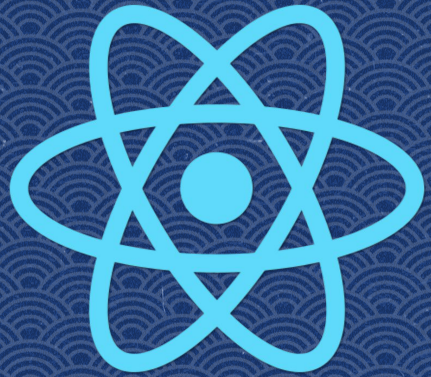
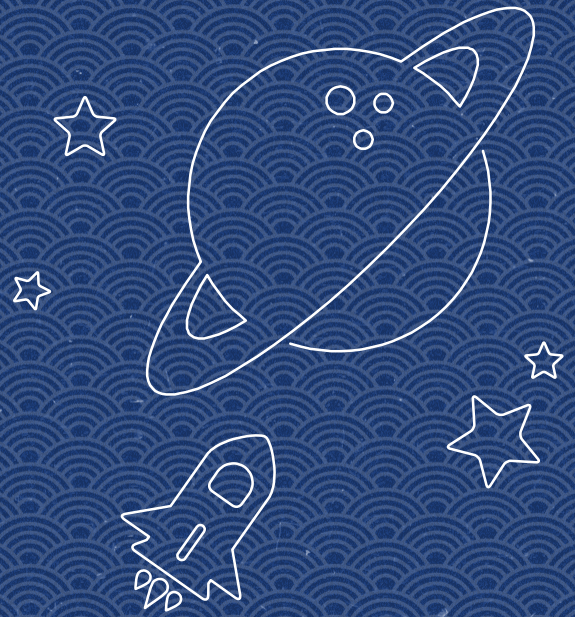
# REACT HOOKS

**Leandro Frigerio**
leandro.frigerio@digitalcareerinstitute.de

*FbW12 - June 2019*





# React Hooks



Hooks are a new addition in React 16.8 (February 2019)

https://github.com/facebook/react/releases/tag/v16.8.0

## WHAT REACT HOOKS ARE

A Hook is a special function that lets you “hook into” React features.

Hooks are a way to use state and other React features without writing a class.

Hooks offer a powerful and expressive new way to reuse functionality between components.

https://reactjs.org/docs/hooks-intro.html

## HOW THEY WORK

"Hooks are a **more direct** way to use the React features you already know — such as state, lifecycle, context, and refs.

They don't fundamentally change how React works, and your knowledge of components, props, and top-down data flow is just as relevant."

## SHOULD I USE HOOKS, CLASSES OR A MIX OF BOTH?

"You can't use Hooks inside of a class component, but you can definitely mix classes and function components with Hooks in a single tree. Whether a component is a class or a function that uses Hooks is an implementation detail of that component. **In the longer term, we expect Hooks to be the primary way people write React components.**"

https://reactjs.org/docs/hooks-faq.html

## HOW THEY WORK

Hooks are JavaScript functions, but you need to follow two rules when using them.

- **Only Call Hooks at the Top Level**

  Don’t call Hooks inside loops, conditions, or nested functions.

- **Only Call Hooks from React Functions**

  Don’t call Hooks from regular JavaScript functions.

https://reactjs.org/docs/hooks-rules.html

## PROBLEMS THE HOOKS ARE TRYING TO ADDRESS

- **Wrapper hell,** Components are surrounded by layers of providers, consumers, higher-order components, render props, and other abstractions
- **Increasing complexity**, something that starts out small becomes large and complex over time, especially as we add lifecycle methods
- **Life cycle methods does too many things**, components might perform some data fetching

## BASIC HOOKS

- **useState**, allows you to use state inside of function component
- **useEffect**, allows you to perform side effect in a way that it replaces several life cycle methods
- **useContext**, accepts a context object (the value returned from React.createContext) and returns the current context value, as given by the nearest context provider for the given context.

## ADDITIONAL HOOKS

- useReducer
- useCallback
- useMemo
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue

https://reactjs.org/docs/hooks-reference.html

## Examples

### With Class

Counter with stateful class component

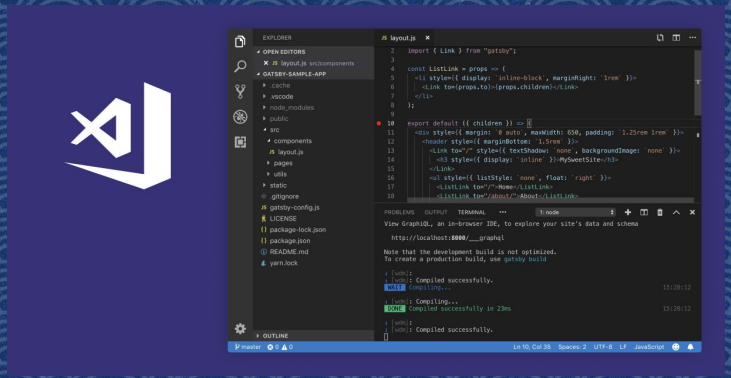
A Component with lifecycle methods

### With Hooks

Counter with functional component and useState

A Component with useEffect

# Live Coding Time!



We start creating a simple Counter using a class Component and refactor it using the Hooks

- We start a new React App
- Create a Counter using a Class Component
- Refactor using some Hooks

# WHAT DAN SAYS

At React Conf 2018, Sophie Alpert and Dan Abramov introduced Hooks, followed by Ryan Florence demonstrating how to refactor an application to use them.

## Resources

- https://reactjs.org/docs/hooks-intro.html
  React Hooks Intro
- https://reactjs.org/docs/hooks-overview.html
  React Hooks Overview
- https://reactjs.org/docs/hooks-reference.html
  React API Reference
- https://www.valentinog.com/blog/hooks/
  React Hook tutorial
- https://usehooks.com/
  Easy to understand React Hook recipes
- https://github.com/rehooks/awesome-react-hooks
  Awesome list of resources


# THANKS!

**Any questions?**

Ask now, in Slack, by email