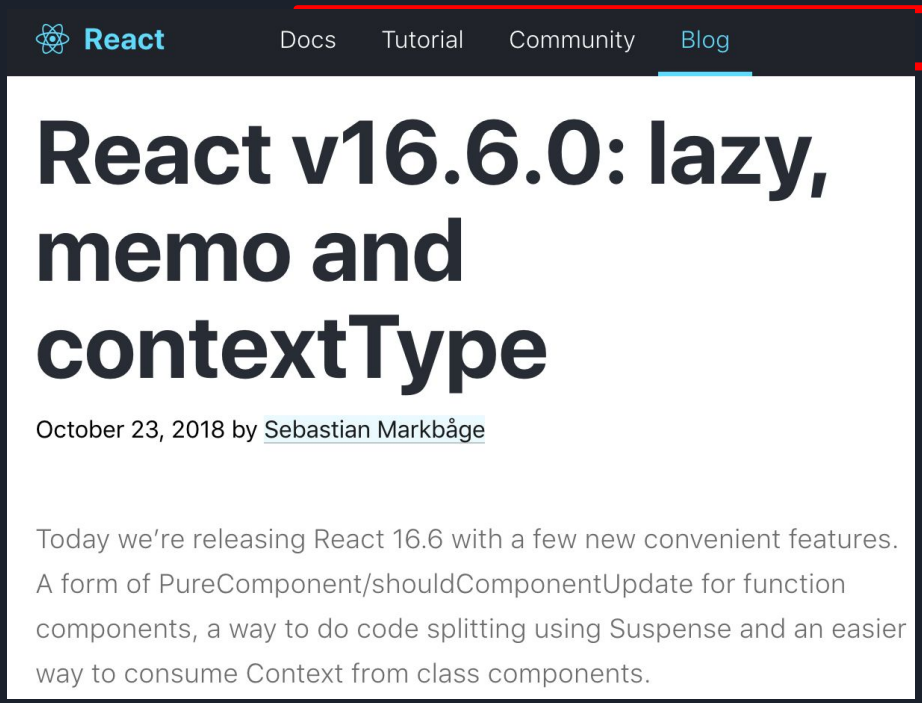
React: displayName

Because trying to debug a nest of Unknown components is kind of meh..

Functional Components

```
const LogoBackdrop = ({ url, color, children }) => {
  const className = 'bc-program-card__logo-backdrop'
  const style = { backgroundColor: color }

  const sharedProps = { className, style }

  if (url) {
    return (
      <a href={url} {...sharedProps} aria-hidden tabIndex="-1">
        {children}
      </a>
    )
  } else {
    return <div {...sharedProps}>{children}</div>
  }
}
```

React 16.6



https://reactjs.org/blog/2018/10/23/react-v-16-6.html

React.memo

Like React.PureComponent, but functional!

React.memo

```
const MyComponent = React.memo(function MyComponent(props) {
  /* render using props */
});
```

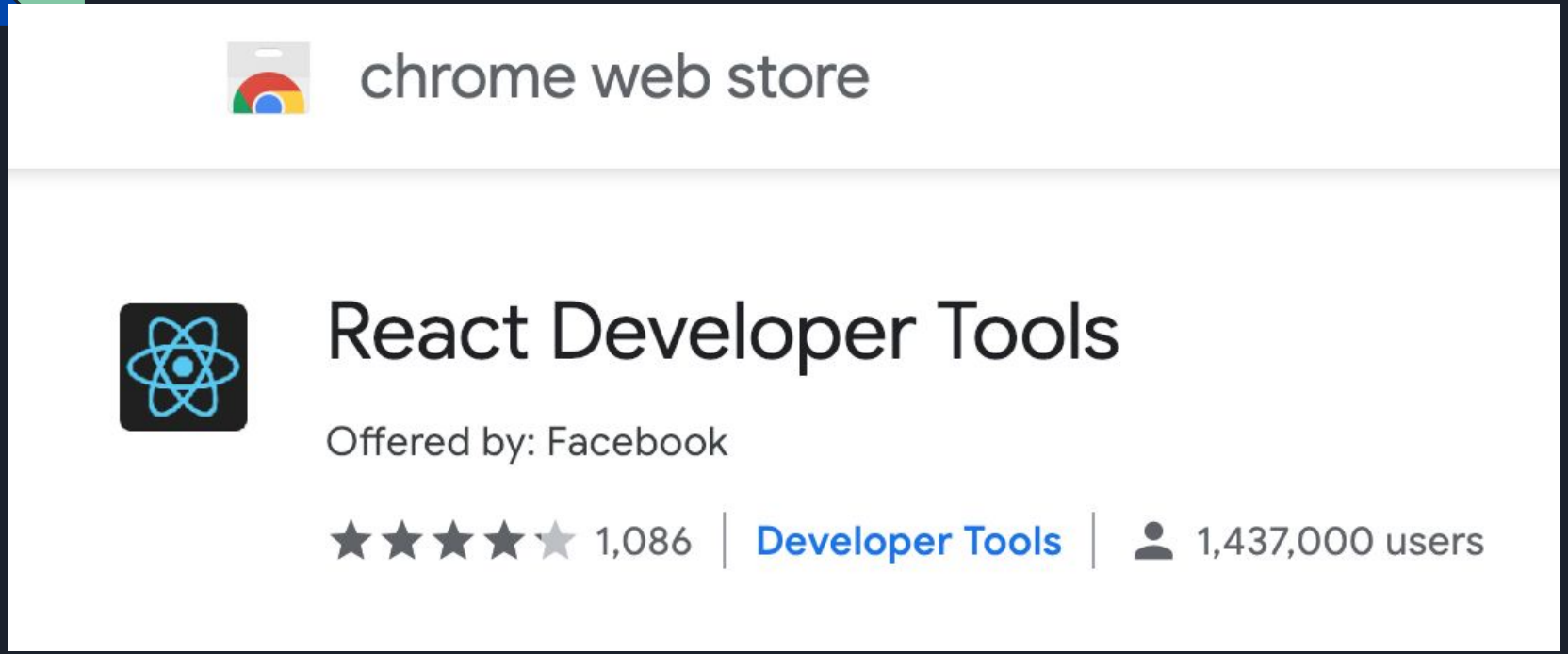
`React.memo` is a higher order component. It's similar to `React.PureComponent` but for function components instead of classes.

https://reactjs.org/docs/react-api.html#reactmemo

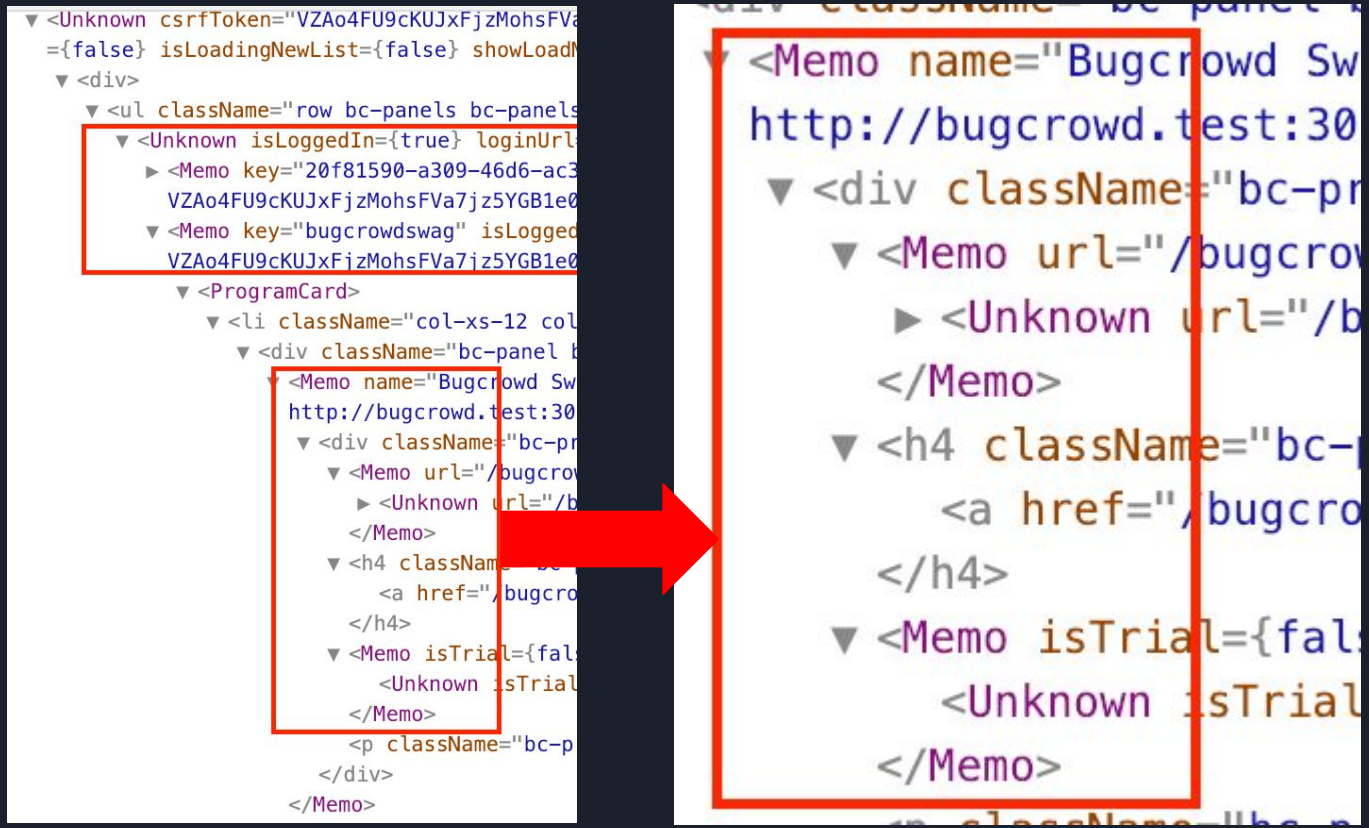
Memoized Functional Components

```
const LogoBackdrop = React.memo( type: ({ url, color, children }) => {
  const className = 'bc-program-card__logo-backdrop'
  const style = { backgroundColor: color }

  const sharedProps = { className, style }

  if (url) {
    return (
      <a href={url} {...sharedProps} aria-hidden tabIndex="-1">
        {children}
      </a>
    )
  } else {
    return <div {...sharedProps}>{children}</div>
  }
})
```

React Developer Tools



Memo -> Memo -> Unknown -> WTF?



displayName?

displayName

The `displayName` string is used in debugging messages. Usually, you don't need to set it explicitly because it's inferred from the name of the function or class that defines the component. You might want to set it explicitly if you want to display a different name for debugging purposes or when you create a higher-order component, see Wrap the Display Name for Easy Debugging for details.

https://reactjs.org/docs/react-component.html#displayname

Maybe not..?

ES6 'Fat Arrow' **anonymous** function

```
const LogoBackdrop = React.memo( type: ({ url, color, children }) => {
  const className = 'bc-program-card__logo-backdrop'
  const style = { backgroundColor: color }
```

Named function

```
const LogoBackdrop = React.memo(function LogoBackdrop({ url, color, children }) {
  const className = 'bc-program-card__logo-backdrop'
  const style = { backgroundColor: color }
```

Wrapping Display Name

Convention: Wrap the Display Name for Easy Debugging

The container components created by HOCs show up in the React Developer Tools like any other component. To ease debugging, choose a display name that communicates that it's the result of a HOC.

The most common technique is to wrap the display name of the wrapped component. So if your higher-order component is named `withSubscription`, and the wrapped component's display name is `CommentList`, use the display name `WithSubscription(CommentList)`:

https://reactjs.org/docs/higher-order-components.html#convention-wrap-the-display-name-for-easy-debugging

memoWithDisplayName

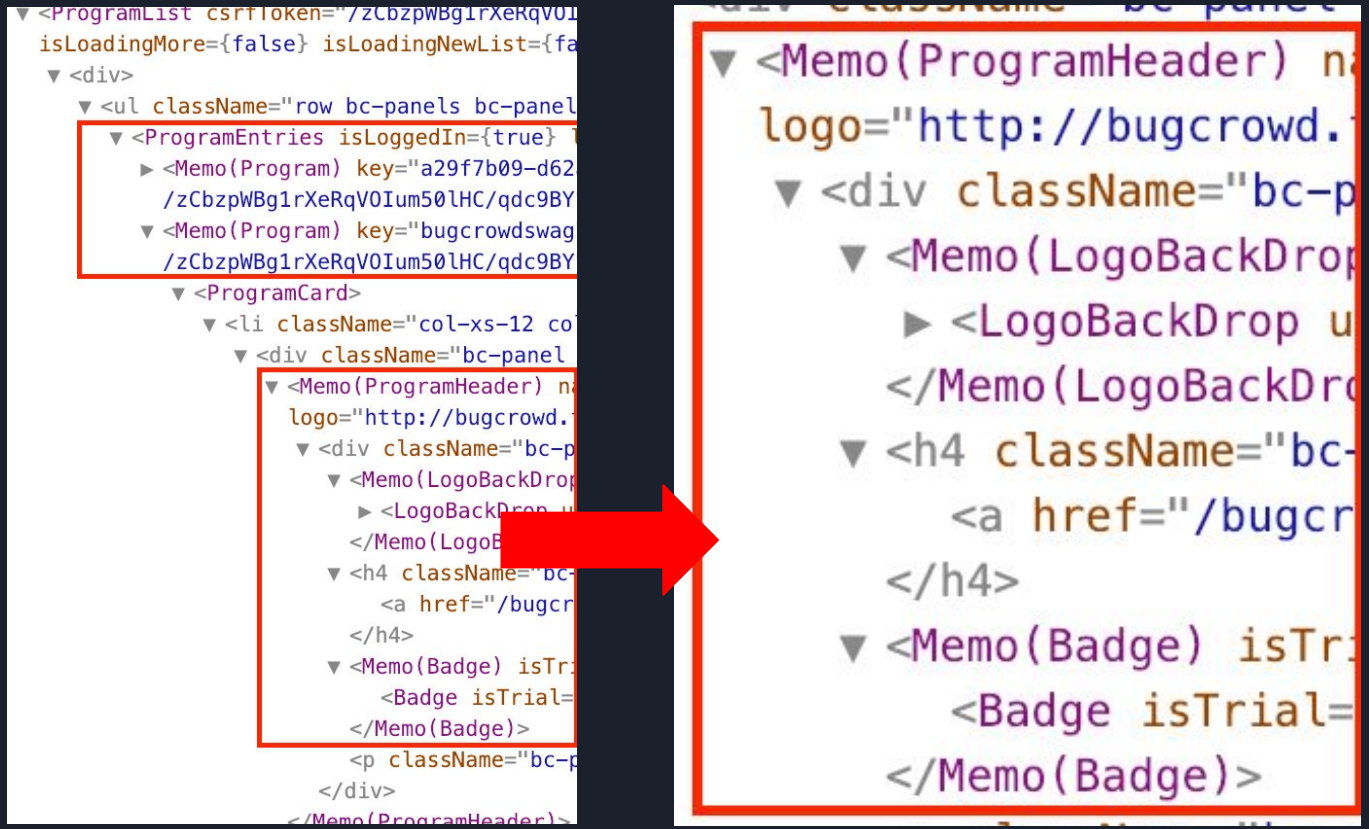
```
/** Explicitly set the displayName on the component that is passed in. ...*/
export const withDisplayName = (displayName, component) => {
  component.displayName = displayName
  return component
}

/** Explicitly set the displayName on the component that is passed in, then wrap
export const memoWithDisplayName = (
  displayName,
  component,
  areEqual = undefined
) => {
  const memoComponent = memo(withDisplayName(displayName, component), areEqual)
  memoComponent.displayName = `Memo(${component.displayName})`
  return memoComponent
}
```

app/javascript/src/shared/react-helpers.js

memoWithDisplayName

```
const LogoBackdrop = memoWithDisplayName(
  displayName: 'LogoBackDrop',
  component: ({ url, color, children }) => {
    const className = 'bc-program-card__logo-backdrop'
    const style = { backgroundColor: color }

    const sharedProps = { className, style }

    if (url) {
      return (
        <a href={url} {...sharedProps} aria-hidden tabIndex="-1">
          {children}
        </a>
      )
    } else {
      return <div {...sharedProps}>{children}</div>
    }
  }
)
```

Components.. withDisplayName(s)!



But how do I know when I need it?

But how do I know when I need it?

But how do I know when I need it?

ESLint: react/display-name

Prevent missing displayName in a React component definition (react/display-name)

DisplayName allows you to name your component. This name is used by React in debugging messages.

Rule Details

The following patterns are considered warnings:

```
var Hello = createReactClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});
```

https://github.com/yannickcr/eslint-plugin-react/blob/master/docs/rules/display-name.md

Thank you!