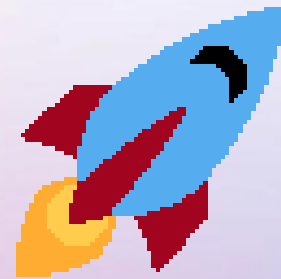


# Unlocking the Mystery of React's Re-Rendering



**Raise your Hand** 🙋

**How many of you think of  
Performance from the beginning?**

## **AFTER-THOUGHT**

**A lot of us think of performance as an after-thought once it becomes a bottleneck**

**or when our app is slow or we find  
a complain in the App Store**



# HI, I'M ANKITA! 🖐️



**Ankita Kulkarni**

Educator | Senior Engineering Leader |  
Developer


# SLIDES & MORE...



Slides and code snippets at:  
[bit.ly/reactrally23](https://bit.ly/reactrally23)

# Why does Performance matter? 🤔



**Your users of the app won't care about which library/framework we use, They care about how fast it loads** 



**You are committing a crime in the tech  
world when your app does not have dark  
mode 😅**

# Let us build a Dark Mode App



***Dark / Light***

Toggle dark mode

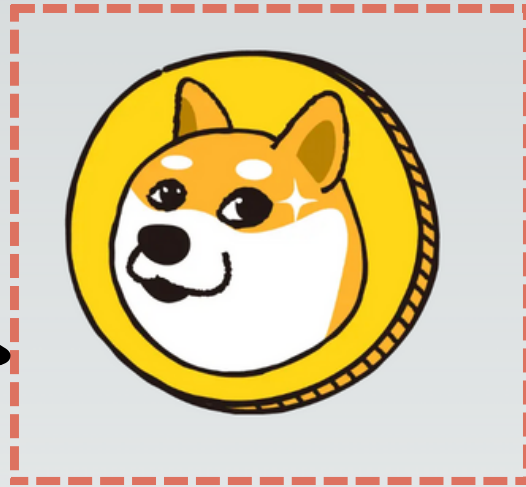


***Dark / Light***

Toggle dark mode

# Let's review the Dark mode app

**Logo**



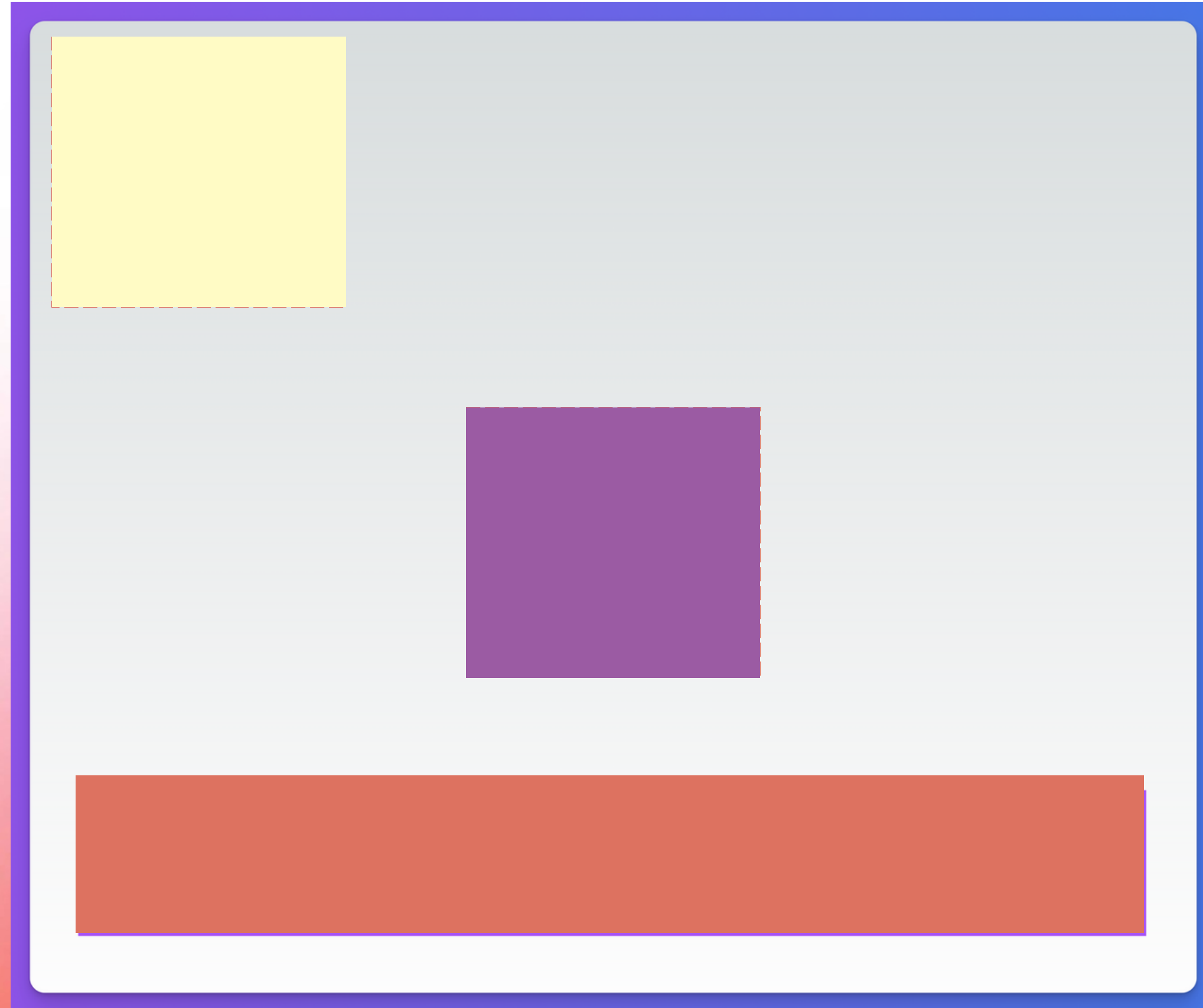
**Toggle Button Handler**



**Footer**

Anky Coby Bean Inc.

# React captures a Snapshot of the darkMode app UI 🎨



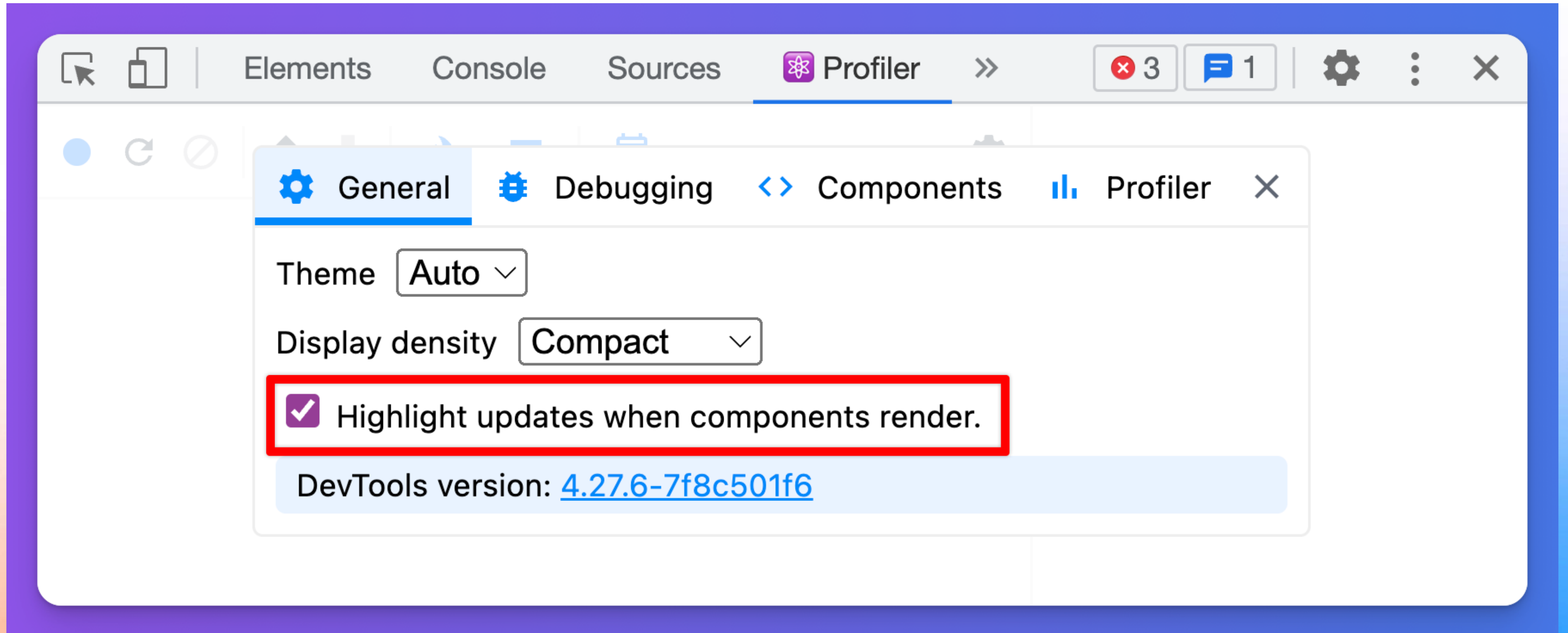


# Think of it like a photographer





# Enable updates in Component Re-renders



# What happens when we click Toggle Dark Mode?

**Logo**



Toggle Dark Mode

Dark mode is disabled.

**Footer**

Anky Coby Bean Inc.

## Code 📌

```
import { useState } from 'react';
import Logo from '@components/logo';
import Footer from '@components/footer';

export default function Home() {
  const [darkMode, setDarkMode] = useState(false);

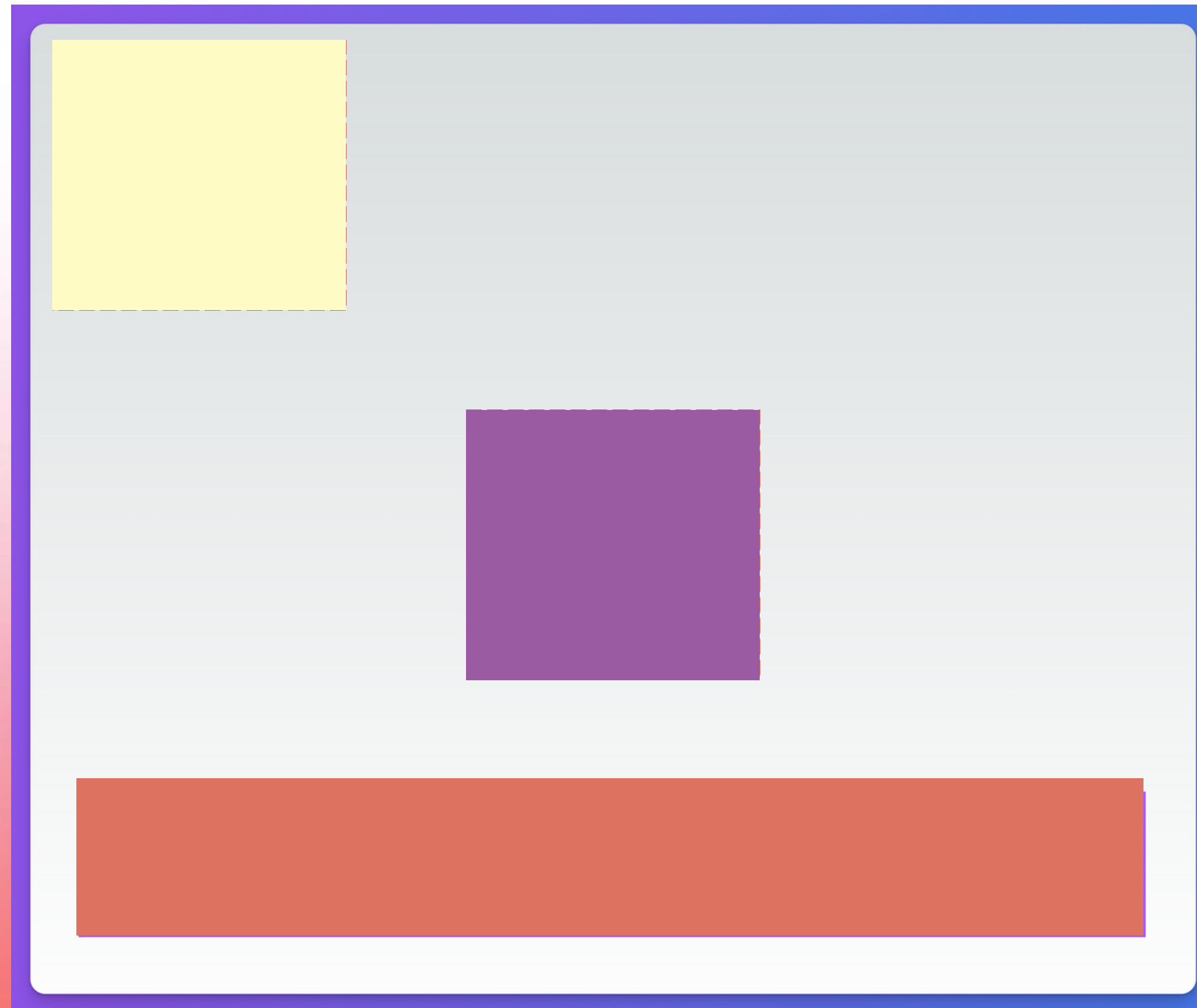
  return (
    <main className={` ${darkMode ? 'dark' : ''} `}>
      <Logo />

      <button onClick={() => setDarkMode(!darkMode)}>
        Toggle Dark Mode
      </button>
      <p>Dark mode is {darkMode ? 'enabled' : 'disabled'}.</p>

      <Footer />
    </main>
  );
}
```

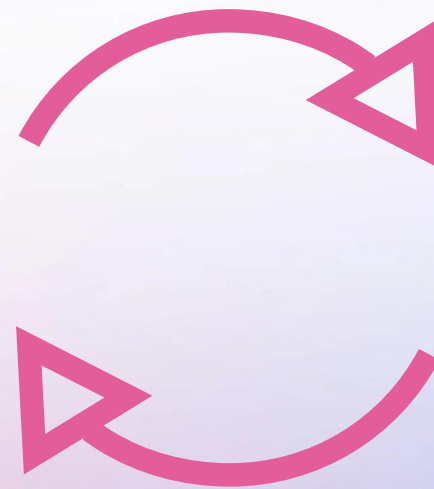


**React re-captures Snapshot of this UI as the state  
darkMode has changed**



# What is Re-rendering?

**Keeping UI  
in sync**



**Application  
state**

# Our Goal with Re-rendering 📖

**REDUCE THE  
AMOUNT OF  
WORK**



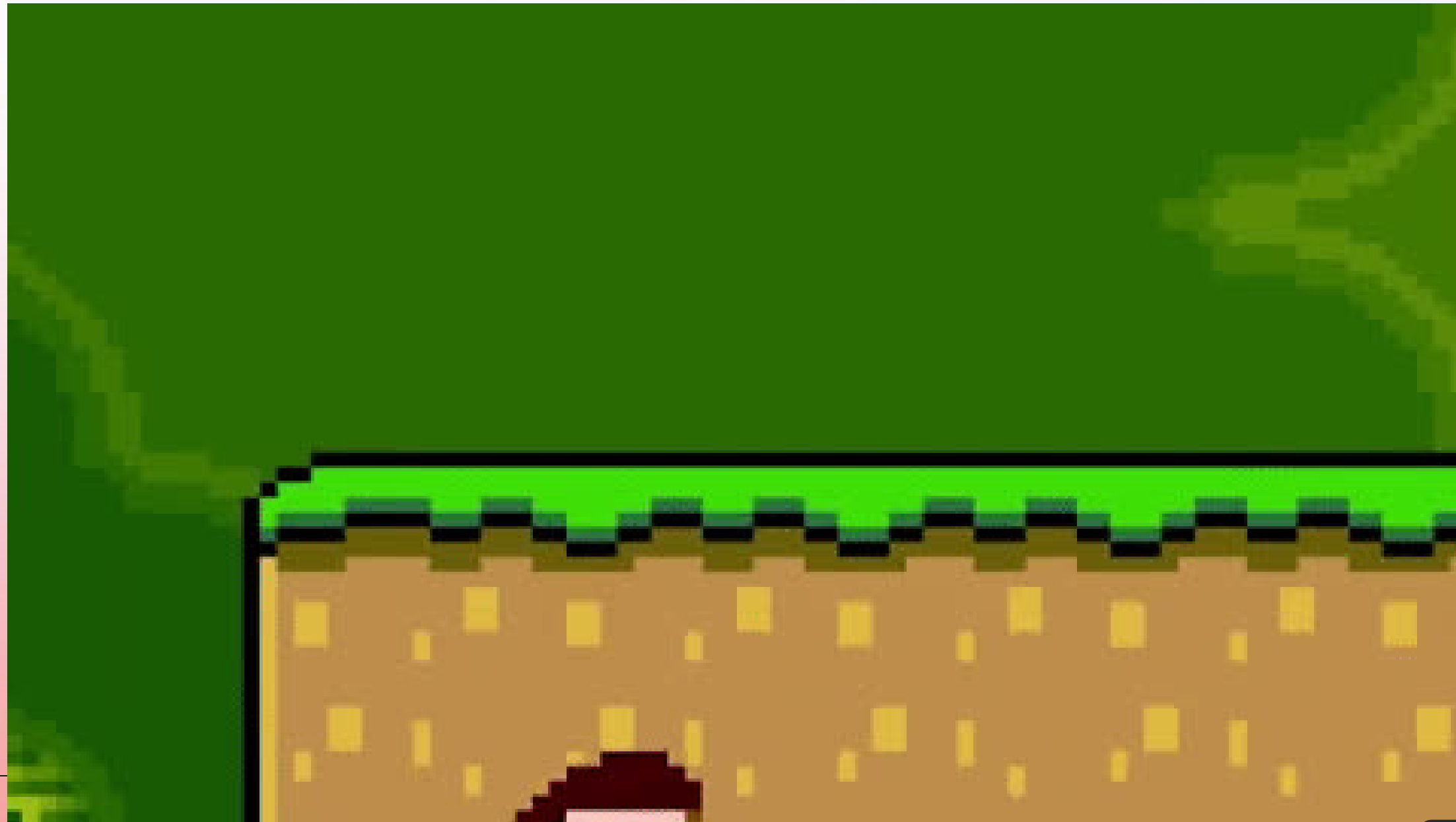
**REDUCE THE  
NUMBER OF  
RE-RENDERS**

**COMMON MISCONCEPTION** 🙄

**Component re-renders due to  
prop changes**



# When a component re-renders, it re-renders all of it's children



**We don't want React to  
re-render when it's not  
needed 👎**

# Why does this happen? 🤔

# How does React work?





# React's UI + React State



# Let's Profile our lil app



Toggle Dark Mode

Dark mode is disabled.

Anky Coby Bean Inc.

# Let's Profile our lil app

Clicking Logo

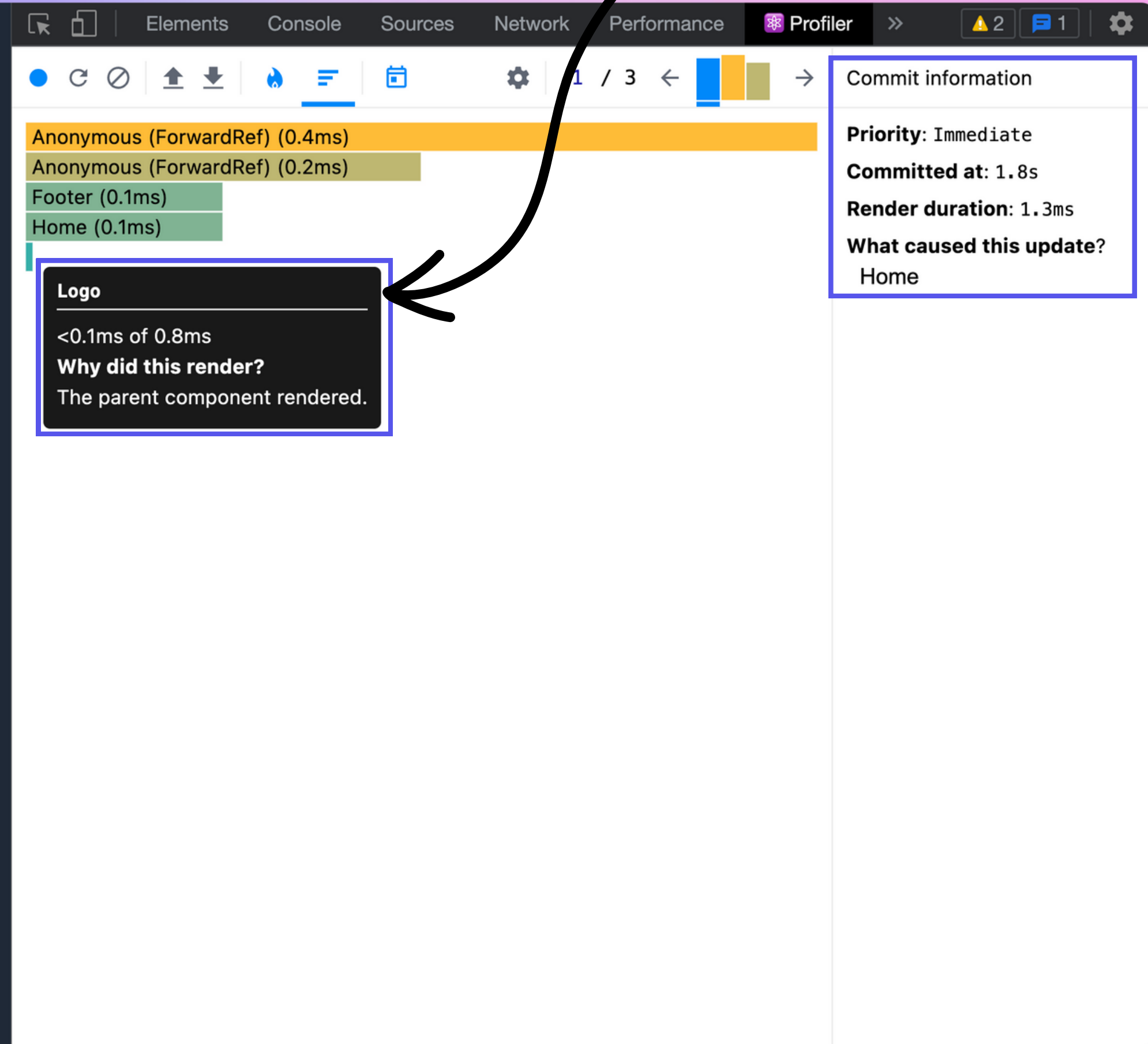


Logo | 756px × 191px

Toggle Dark Mode

Dark mode is enabled.

Anky Coby Bean Inc.



# How do we solve for this?



# Let's wrap Logo & Footer in memo

```
import Image from 'next/image';
import React, { memo } from 'react';

const Logo = () => {
  return (
    <div className="hover:cursor-pointer">
      <Image alt="doge" src="/doge.webp" width={'200'} height={'200'} />
    </div>
  );
};

export default memo(Logo);
```

# Introducing Memoization 🔥

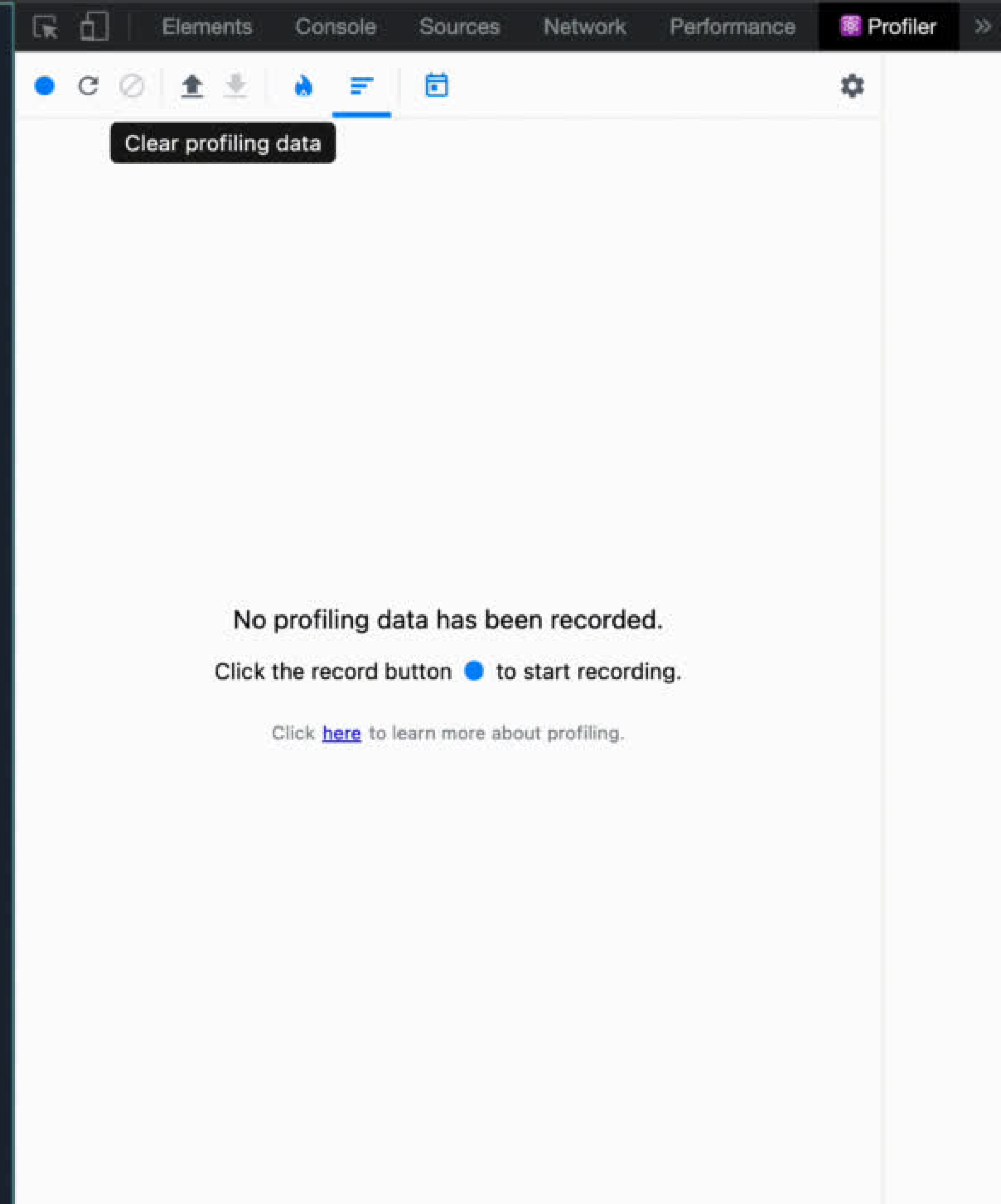
# Let's Profile our lil app again



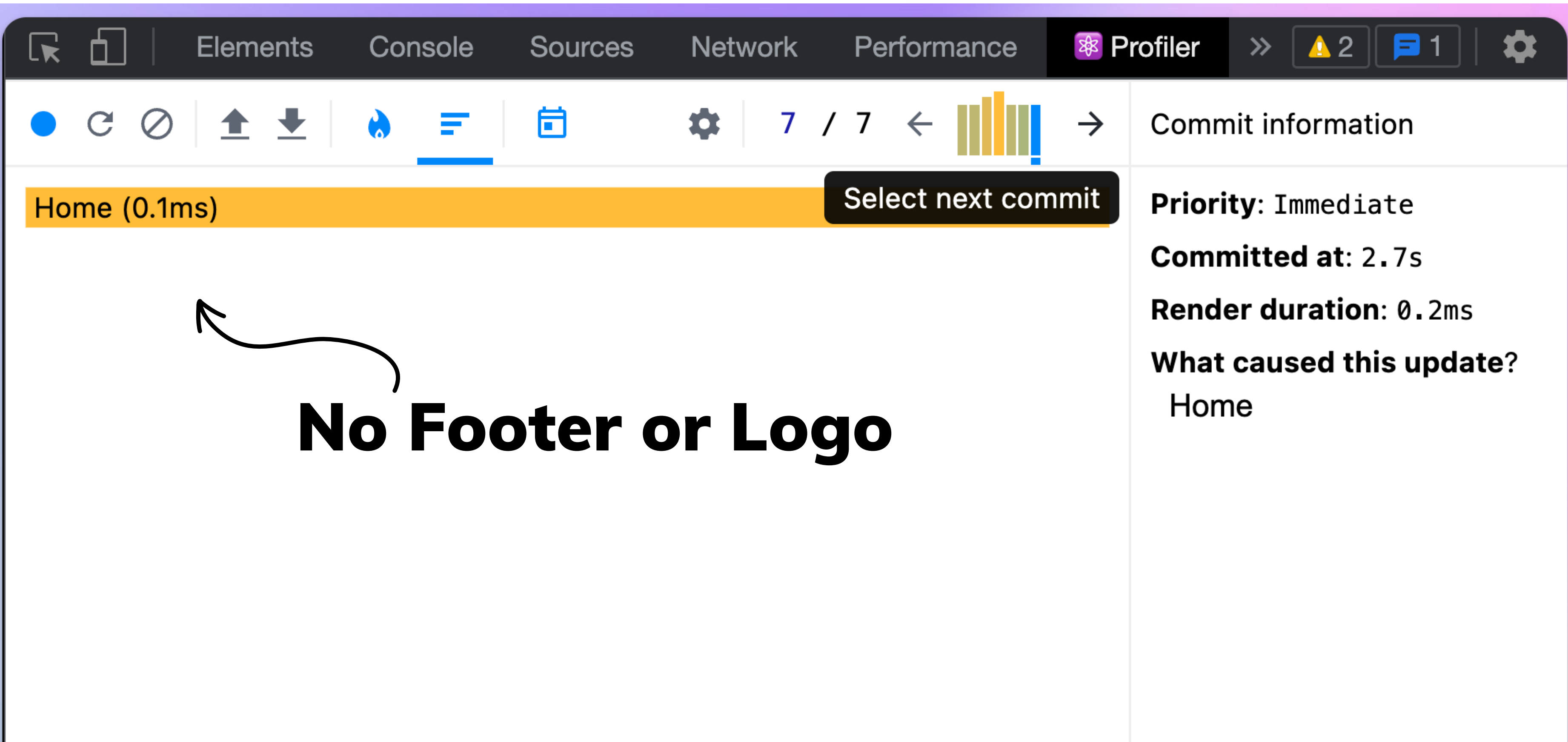
Toggle Dark Mode

Dark mode is enabled.

Anky Coby Bean Inc.



# If we review it



The screenshot shows the Chrome DevTools Profiler interface. The top toolbar includes tabs for Elements, Console, Sources, Network, Performance, and Profiler. The Profiler tab is active, displaying a commit bar for 'Home' with a duration of 0.1ms. A hand-drawn arrow points from the text 'No Footer or Logo' to the commit bar. The right sidebar shows commit information:

- Commit information
- Priority: Immediate
- Committed at: 2.7s
- Render duration: 0.2ms
- What caused this update?  
Home



# Dark Mode Code

```
import { useState } from 'react';
import Logo from '@components/logo';
import Footer from '@components/footer';

export default function Home() {
  const [darkMode, setDarkMode] = useState(false);

  return (
    <main className={` ${darkMode ? 'dark' : ''} `}>
      <Logo />

      <button onClick={() => setDarkMode(!darkMode)}>
        Toggle Dark Mode
      </button>
      <p>Dark mode is {darkMode ? 'enabled' : 'disabled'}.</p>
      <Footer />
    </main>
  );
}
```

# Let's move Dark Mode logic outside

# useDarkMode hook 🔥

```
1 import { useCallback, useState } from 'react';
2
3 const useDarkMode = () => {
4   const [darkMode, setDarkMode] = useState(false);
5
6   const handleToggle = useCallback(() => setDarkMode(!darkMode), [darkMode]);
7
8   return [darkMode, handleToggle];
9 };
10
11 export default useDarkMode;
```

# useCallback

**CACHES** FUNCTIONS

**RETURNS  
CACHED  
FUNCTION**



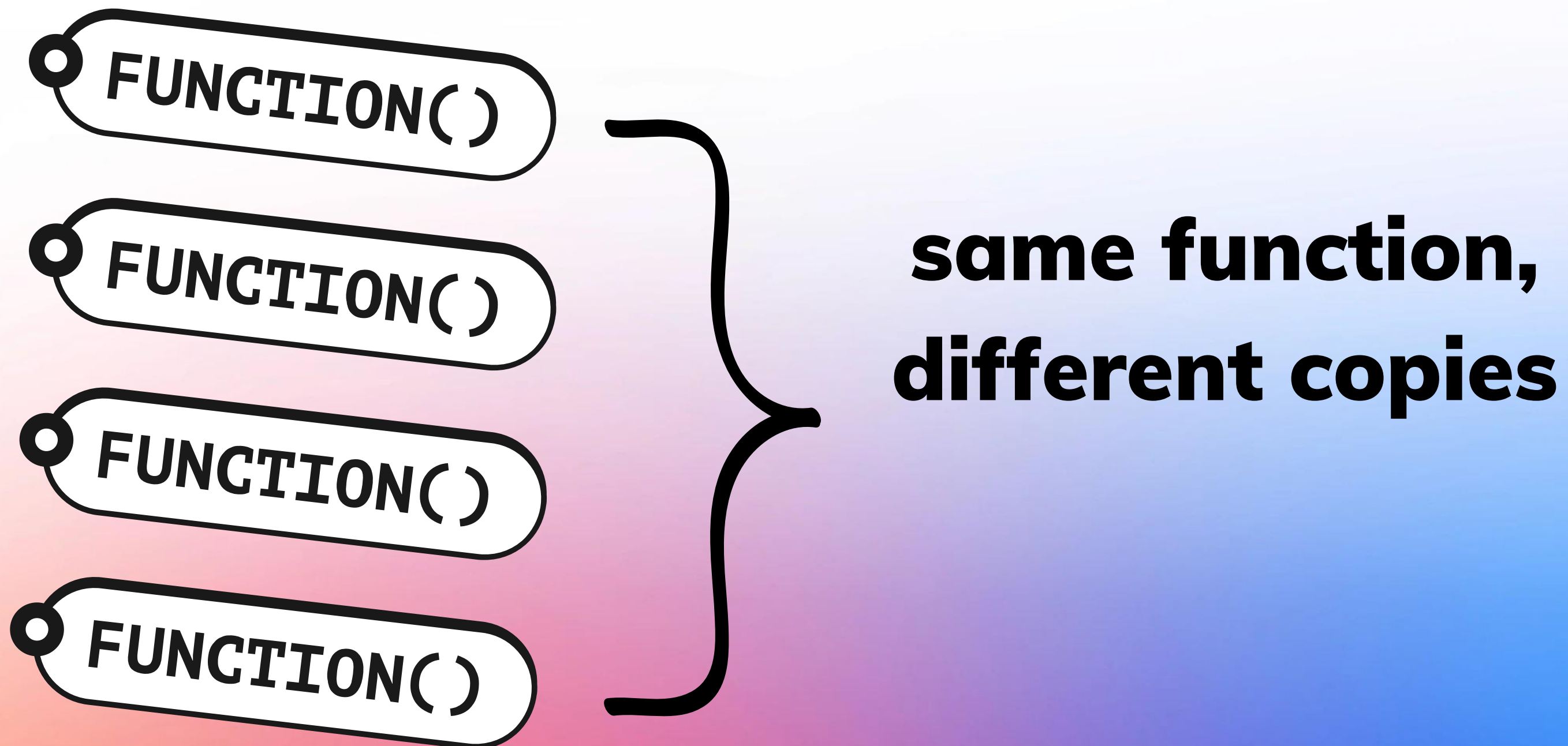
**[]  
DEPENDANCY  
ARRAY  
PURGES CACHE**



# useDarkMode hook 🔥

```
1 import { useCallback, useState } from 'react';
2
3 const useDarkMode = () => {
4   const [darkMode, setDarkMode] = useState(false);
5
6   const handleToggle = useCallback(() => setDarkMode(!darkMode), [darkMode]);
7
8   return [darkMode, handleToggle];
9 };
10
11 export default useDarkMode;
```

# useCallback: dependancies change



# useMemo

**CACHES THINGS**



**RETURNS  
CACHED VALUE**

**[]  
DEPENDANCY  
ARRAY  
PURGES CACHE**



## useMemo

Sets colour  
to blue on initialization

This fn  
increment the  
sum by 1,  
100000000 times

```
import React, { useState, useMemo } from 'react';

function App() {
  const [color, setColor] = useState('blue');

  const expensiveComputation = useMemo(() => {
    console.log('Expensive computation running...');
    let sum = 0;
    for (let i = 0; i < 100000000; i++) {
      sum += i;
    }
    return sum;
  }, [color]);

  return (
    <div>
      <h2>Color: {color}</h2>
      <h3>Expensive Computation: {expensiveComputation}</h3>
      <button onClick={() => setColor(color === 'blue' ? 'red' : 'blue')}>
        Change Color
      </button>
    </div>
  );
}

export default App;
```

With useMemo, this fn  
would run only when  
state color changes

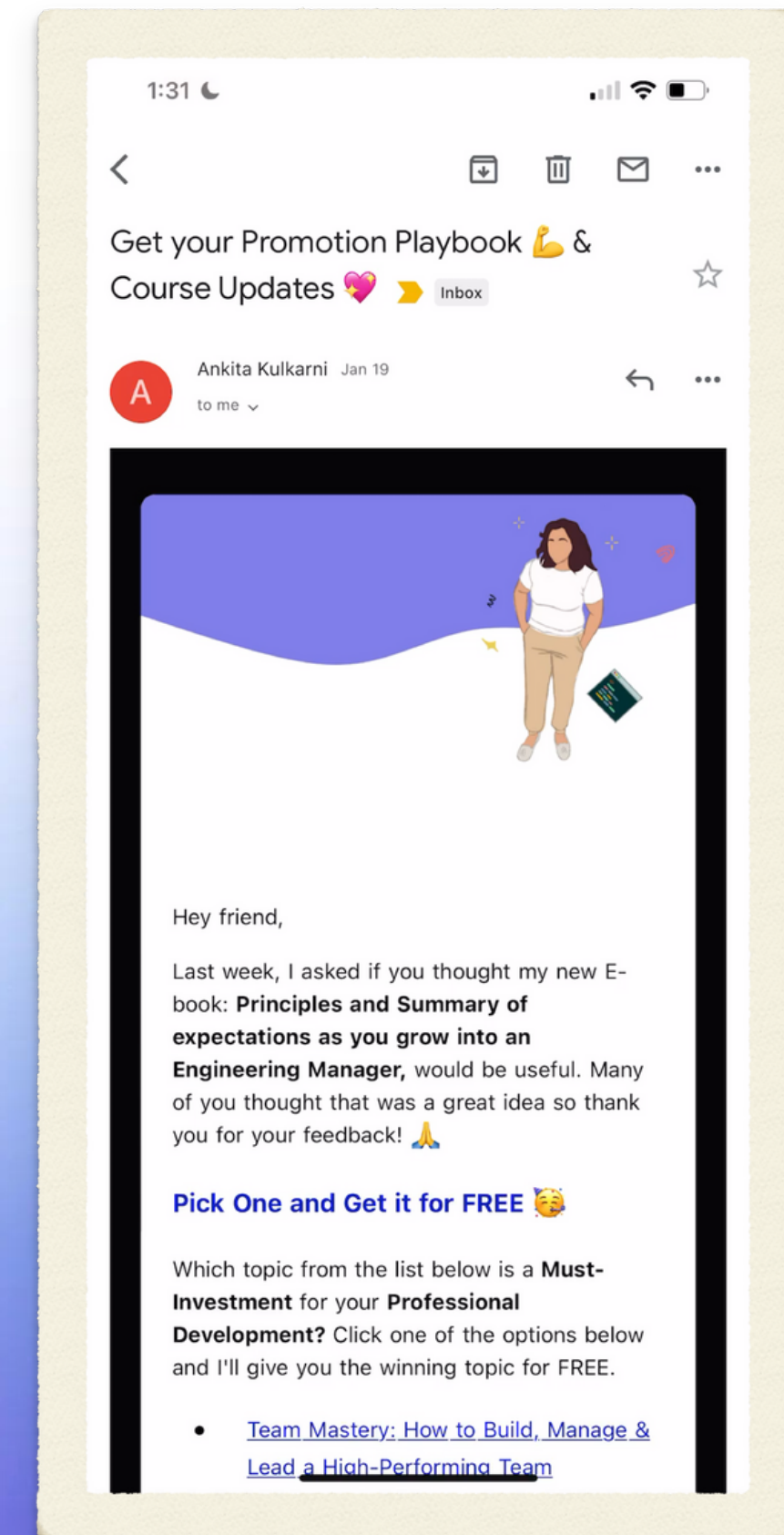
Without useMemo, this fn  
would run every time



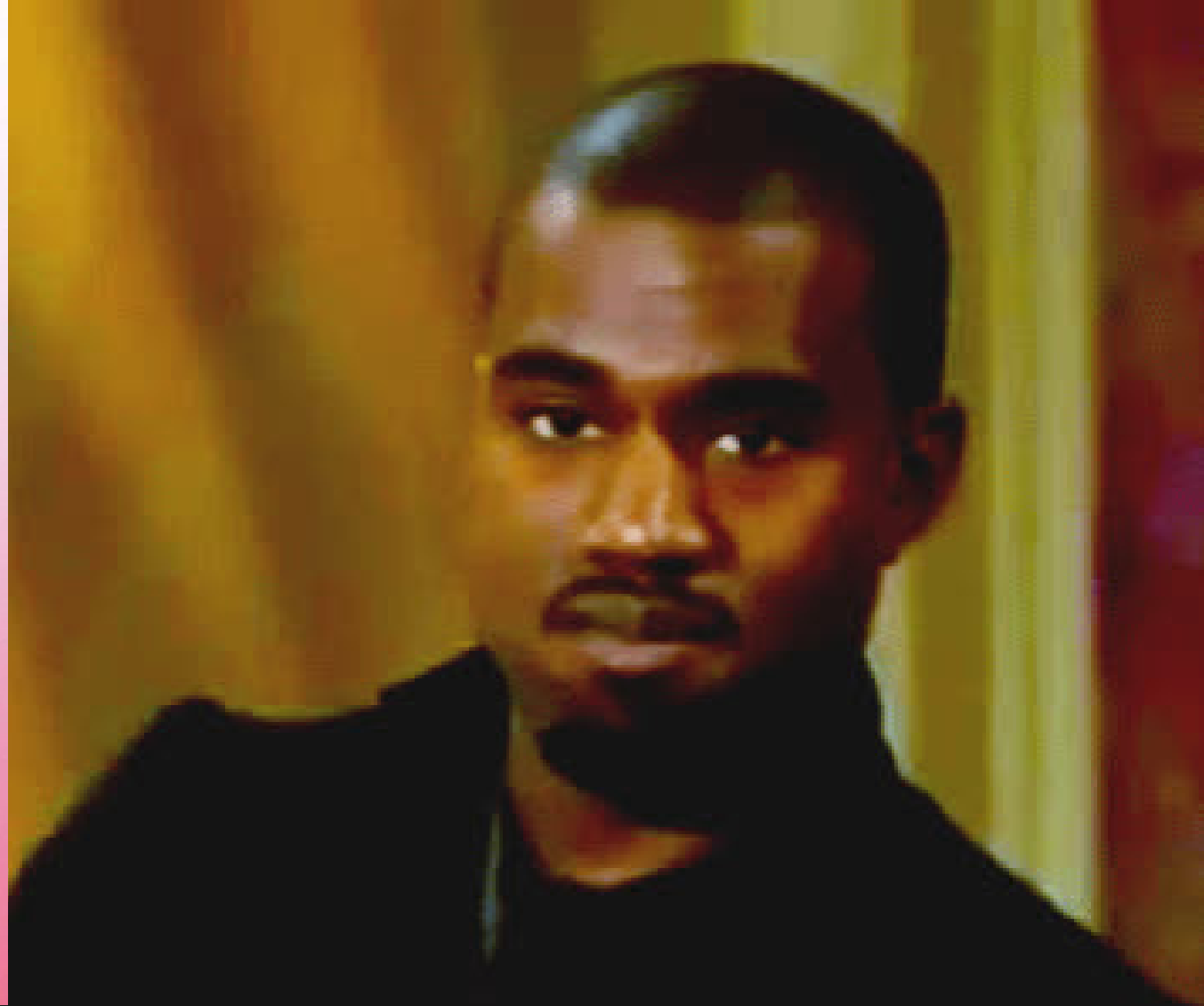
# Weekly Newsletter

Grab your weekly  
Frontend or Leadership  
Snacks 🍿

[kulkarniankita.com/newsletter](https://kulkarniankita.com/newsletter)



**I'm not proposing to use  
useCallback & useMemo like  
hot cakes**



# Performance Tips ✨





# Test it on every device and network

A lot of us have high-speed internet but that doesn't represent our users



# Things to focus on

01

## Test on low-end devices

Try using older devices not just iphones

02

## Test on slower networks and Throttle

03

## Profile your app on Prod mode, not just dev mode

Dev mode doesn't represent real use case

# Don't try to fix what's not broken



# Again, Let's remember our Goal with Re-rendering 📖

**REDUCE THE  
AMOUNT OF  
WORK**



**REDUCE THE  
NUMBER OF  
RE-RENDERS**

# **Should you use useMemo & useCallback everywhere?**

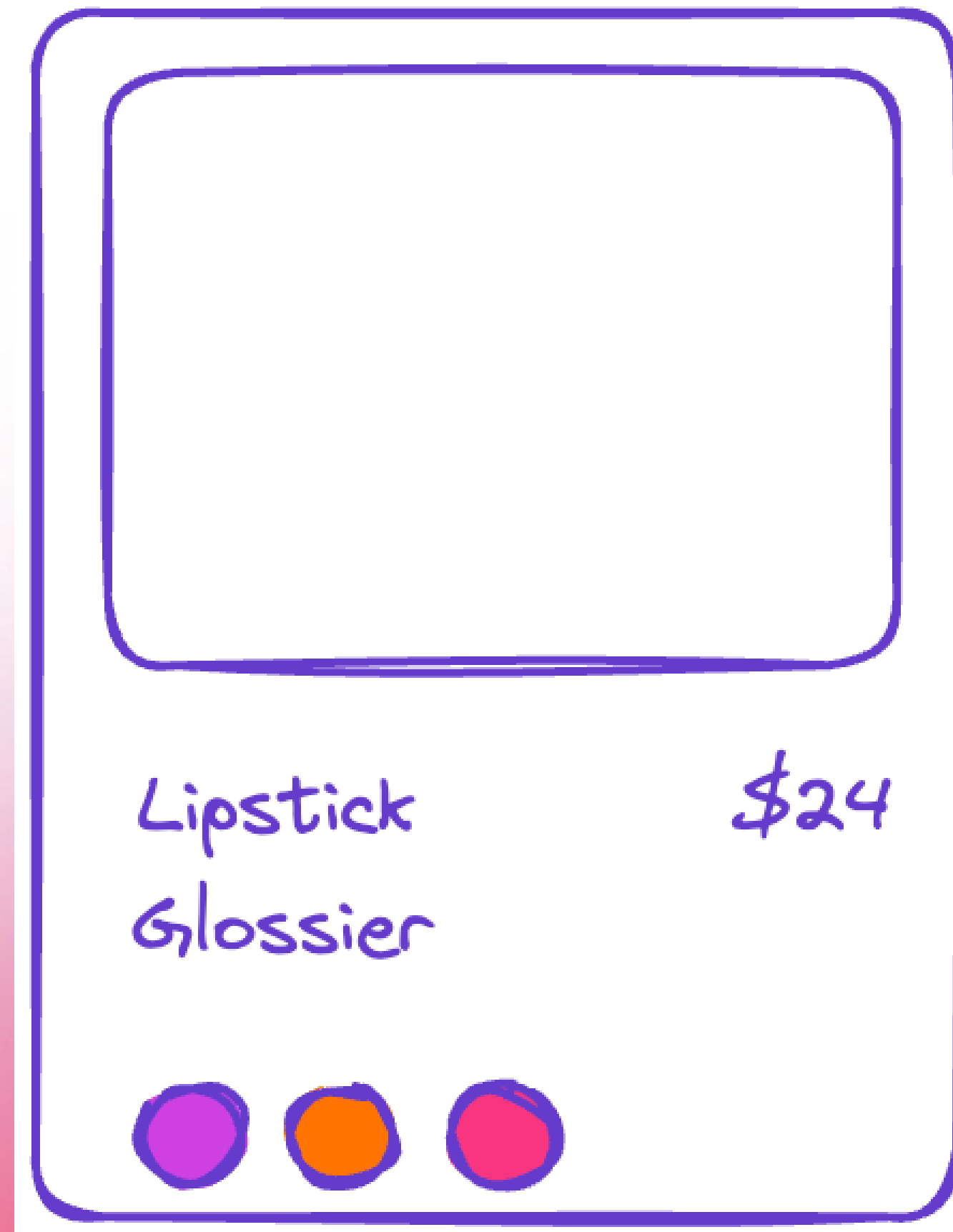


# Should you use useMemo & useCallback everywhere?



**How do we do this for Apps  
with hundreds of  
Components?**

# Complex Component 📌



Do you want to  
wrap every fn  
in useCallback  
or useMemo?





It will be really  
expensive 😅



# Lighthouse doesn't fully represent performance ⚡

# FREE mini course on React Performance

[bit.ly/reactperformance23](https://bit.ly/reactperformance23)



# Thank you! 🙌

More resources at [kulkarniankita.com](https://kulkarniankita.com)

All my slides are at [bit.ly/reactrally23](https://bit.ly/reactrally23)

