

Performance Myths of React Native Apps

Checklist for a Great Talk



While waiting for others to come in, here are some rules and reminders to keep in mind.

01

Nod 'yes' when you agree with me

02

Have questions? come find me after

03

Want more ENERGY ⚡ in the room

Raise your Hand 🖐️

**How many of you believe that
React Native is slower than native
app development?**

Raise your Hand 🖐️

**How many of you want to say
"it depends" to the previous
question?**

HI, I'M ANKITA!



Ankita Kulkarni

Educator | Senior Engineering Leader |
Developer

SLIDES & MORE...

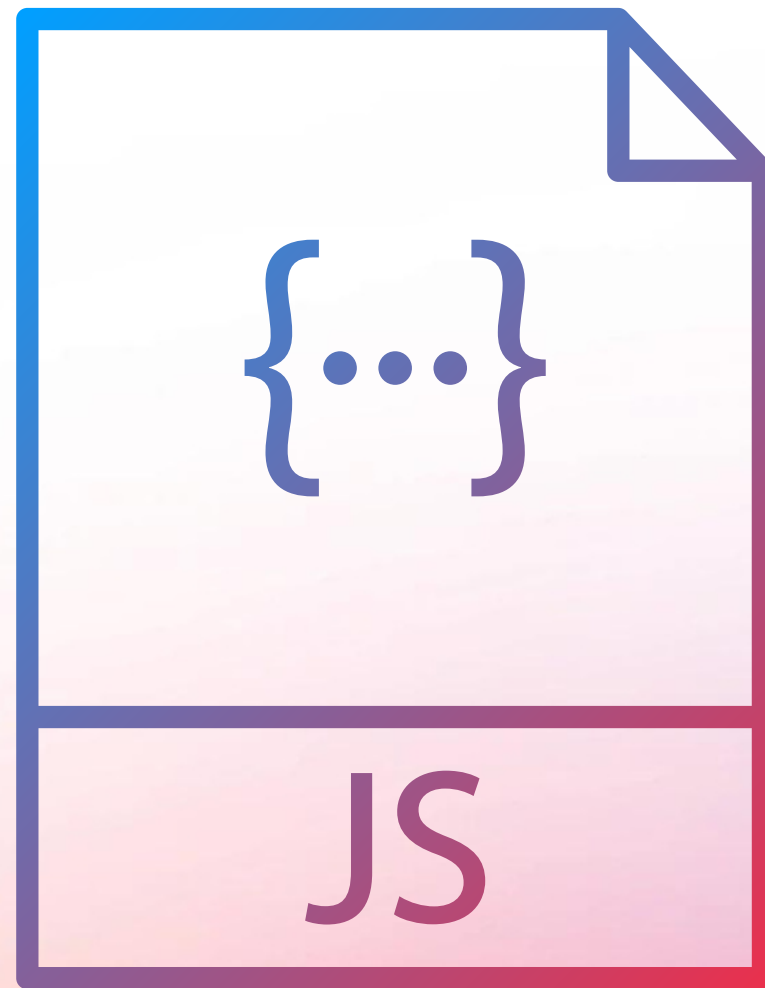


Slides and code snippets at:
bit.ly/appjs23

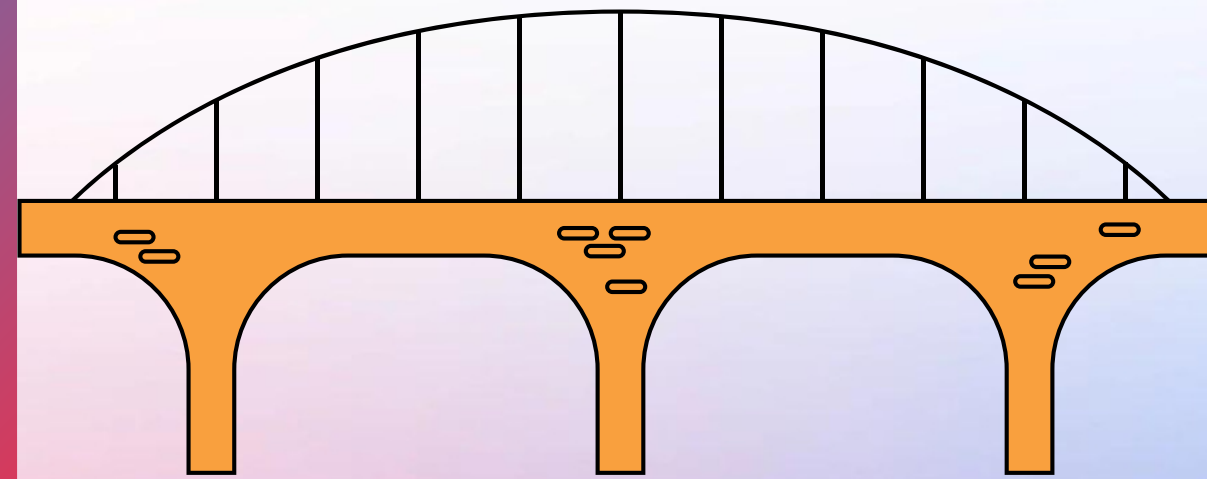
Weekly Newsletter

**Grab your weekly
Frontend or Leadership
Snacks**

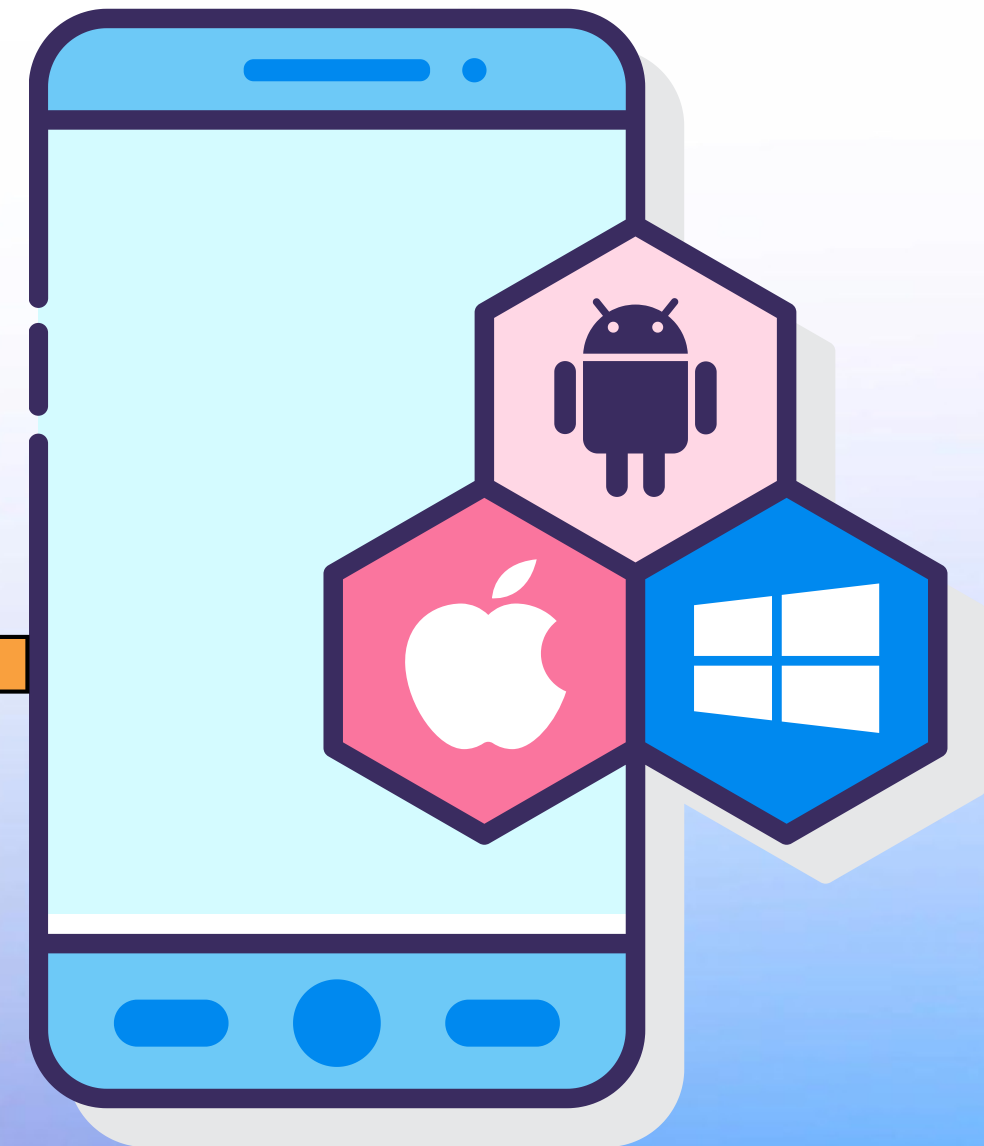
kulkarniankita.com/newsletter



JavaScript



Bridge

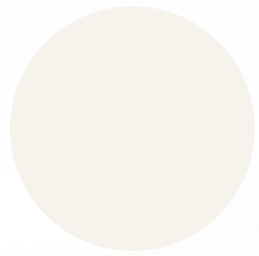


Mobile

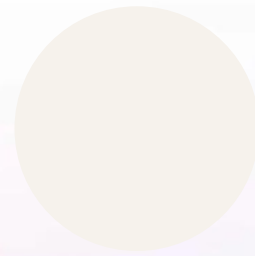
Quick story

Beers with friends

Objections



**JavaScript is not
enough**



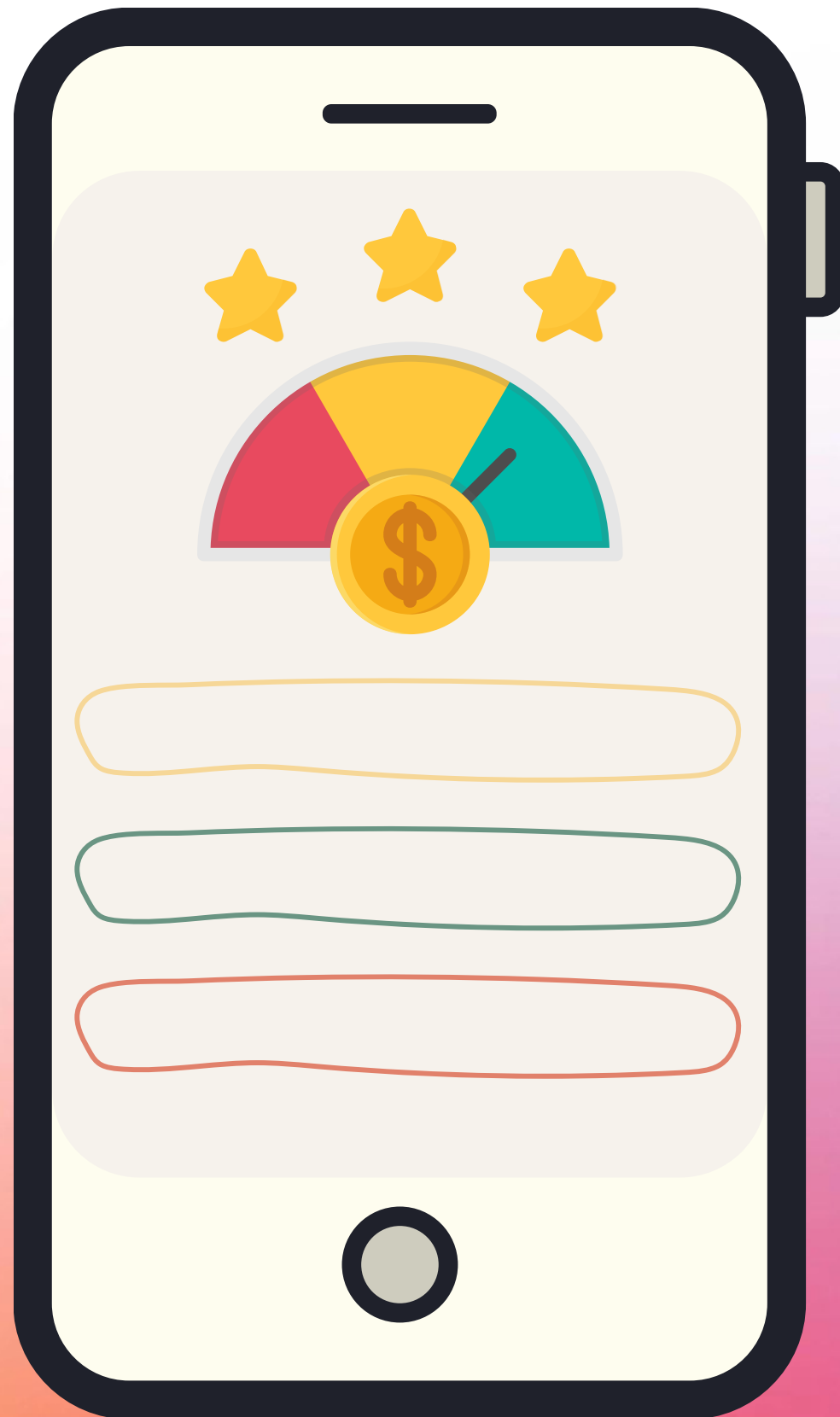
**You can't animate
correctly**



**Performance Issues
such as high cpu etc.**

Let's address these + more

Let's build an app together: Let's spend your Money



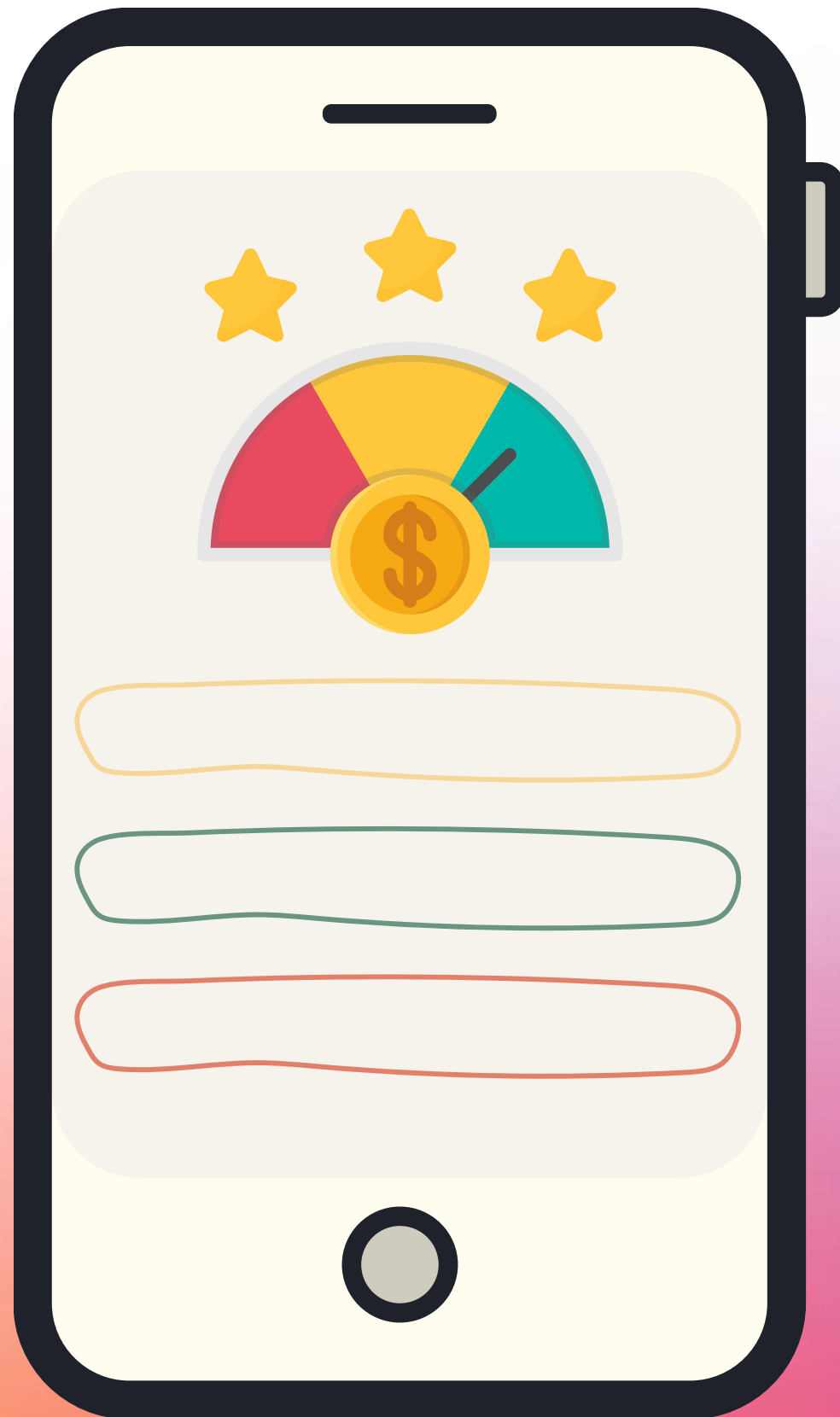
Our criteria

Functional

Is fast to load

Has cool Animations
should work low data

Criteria for Measuring Performance



Measure a specific behaviour

Keep the same set of test cases

Disable Dev Mode and test in Prod Mode

3 important criteria for Performance

01

CPU & Memory

Less CPU

02

Application Size

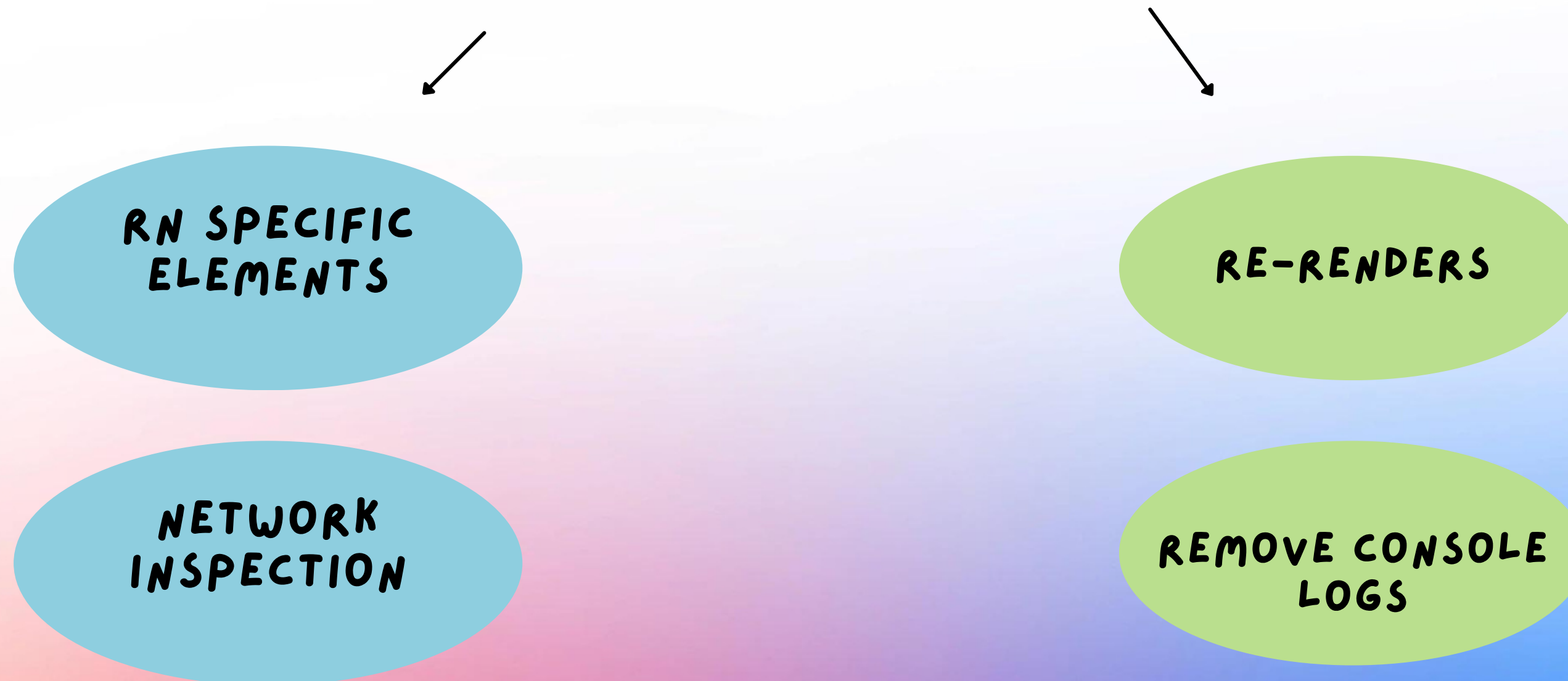
Download quickly and smaller app

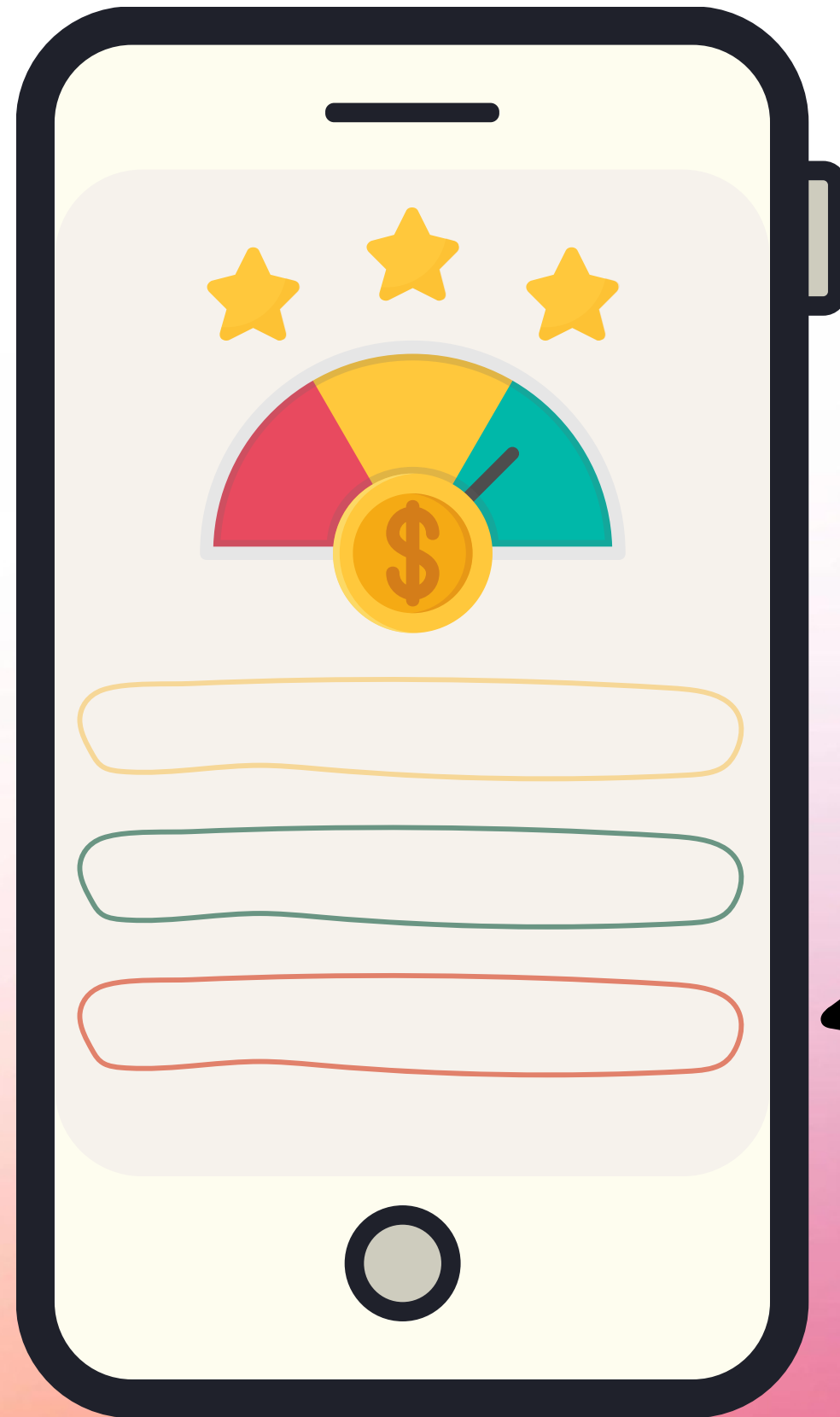
03

App Startup time and Frames Per Second

it has to run at 60fps else it's slow

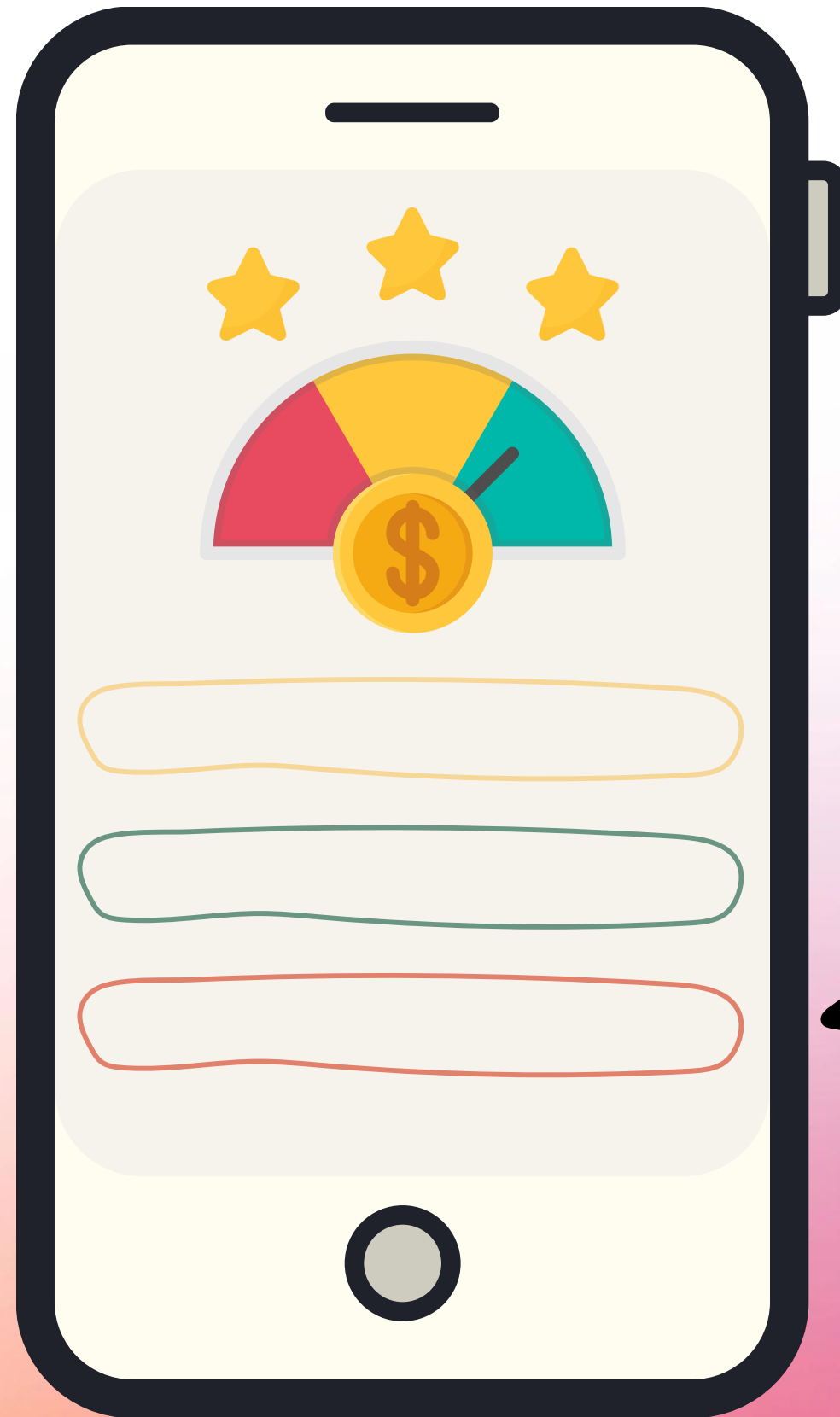
We can approach this using 2 ways





Use FlatList and Section List





**As you scroll,
the items re-render**

Introducing Memoization

Wrap the component in memo(Component)

Don't try to fix what's not broken



Our Goal with Re-rendering

**REDUCE THE
AMOUNT OF
WORK**



**REDUCE THE
NUMBER OF
RE-RENDERS**

Should you use `useMemo` & `useCallback` everywhere?

`useMemo(() => fn, [])`

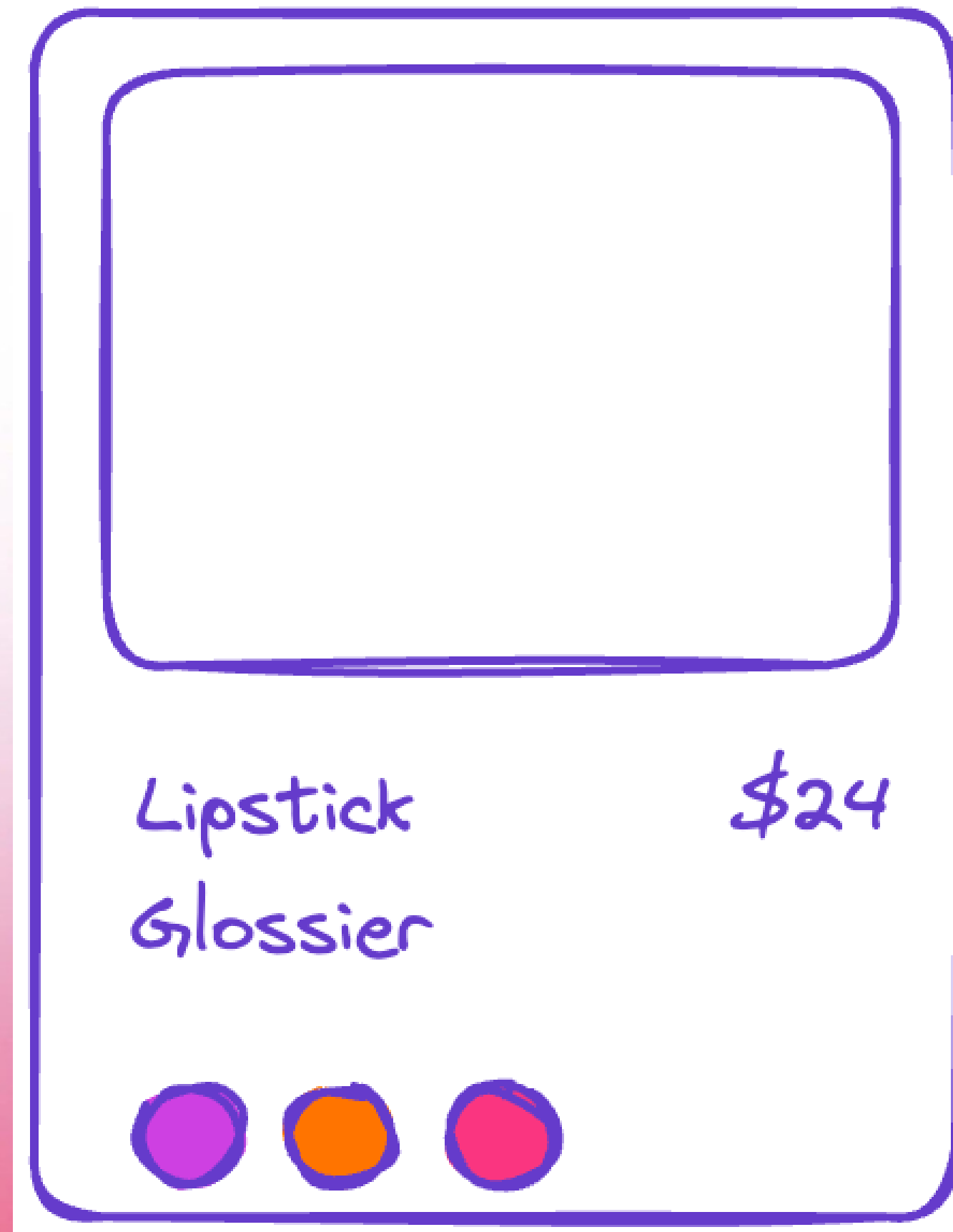
`dependencies`



Calculation of Props is more costly than the entire component re-render



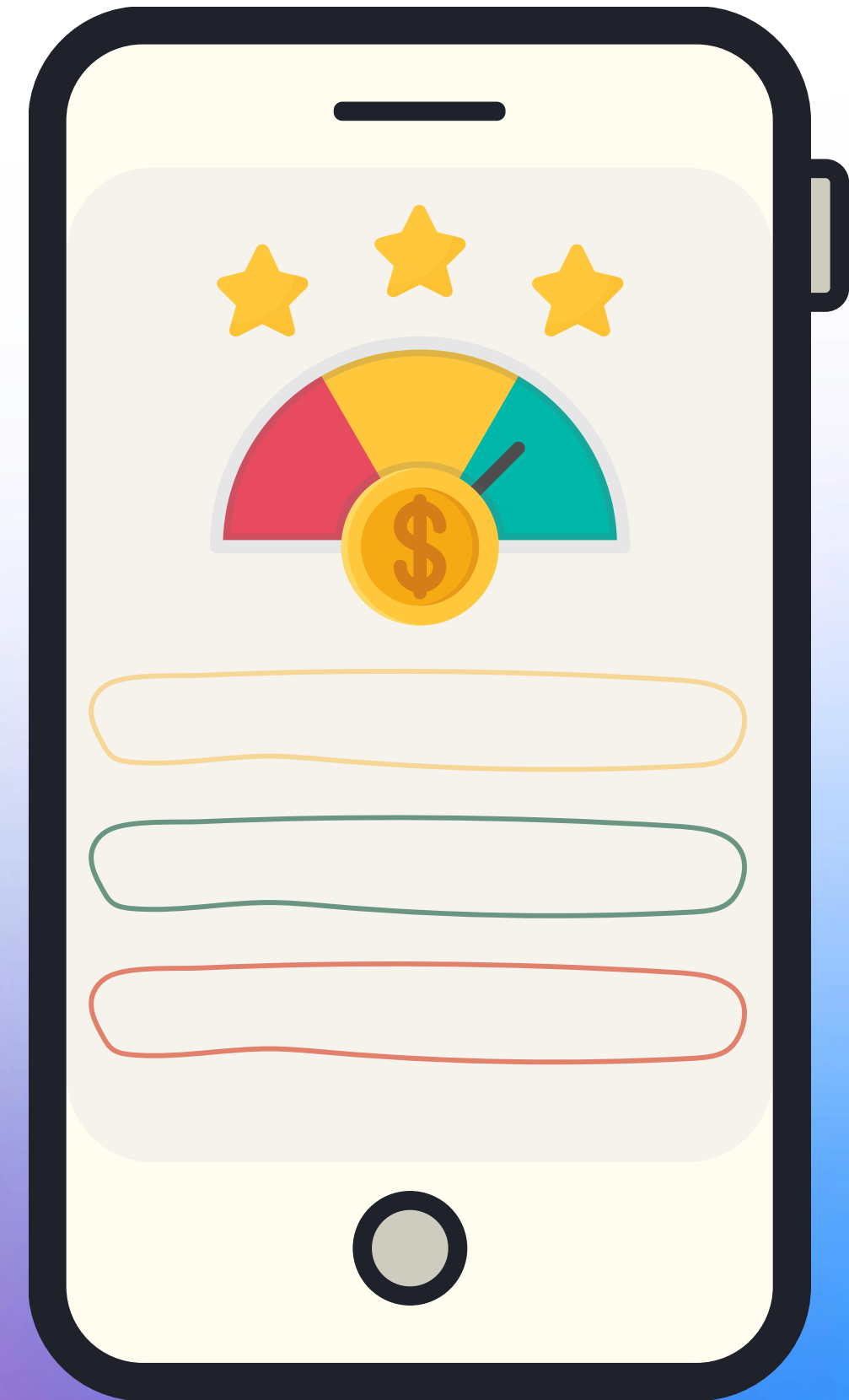
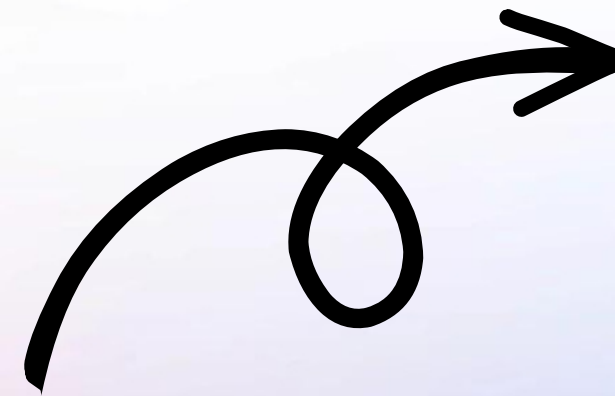
Complex Component



Remove unnecessary console logs



Let's implement Animation and Chart



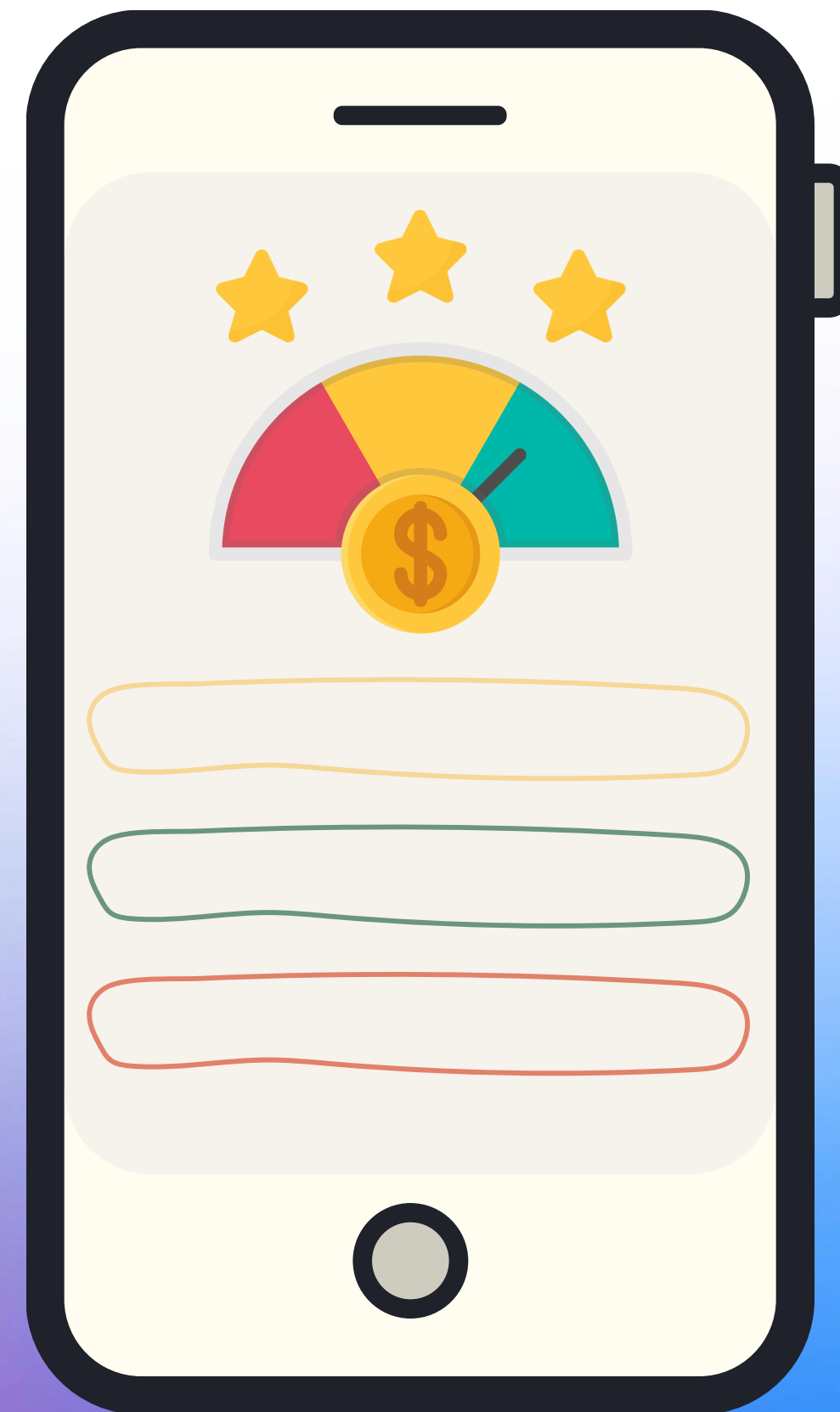
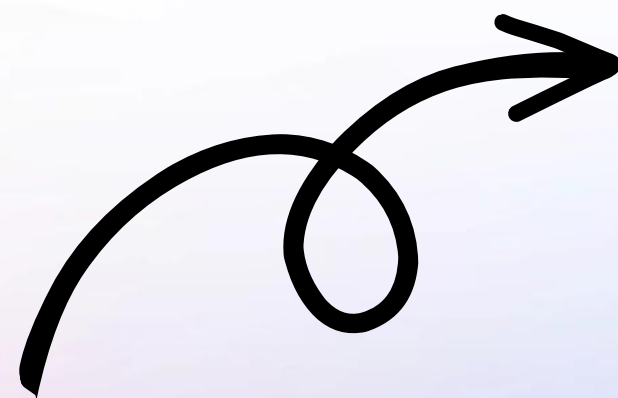
MYTH BUSTER

**React Native cannot handle
complex Animations**

60 frames per second (fps)



Let's use Reanimated



Be careful of your library choice

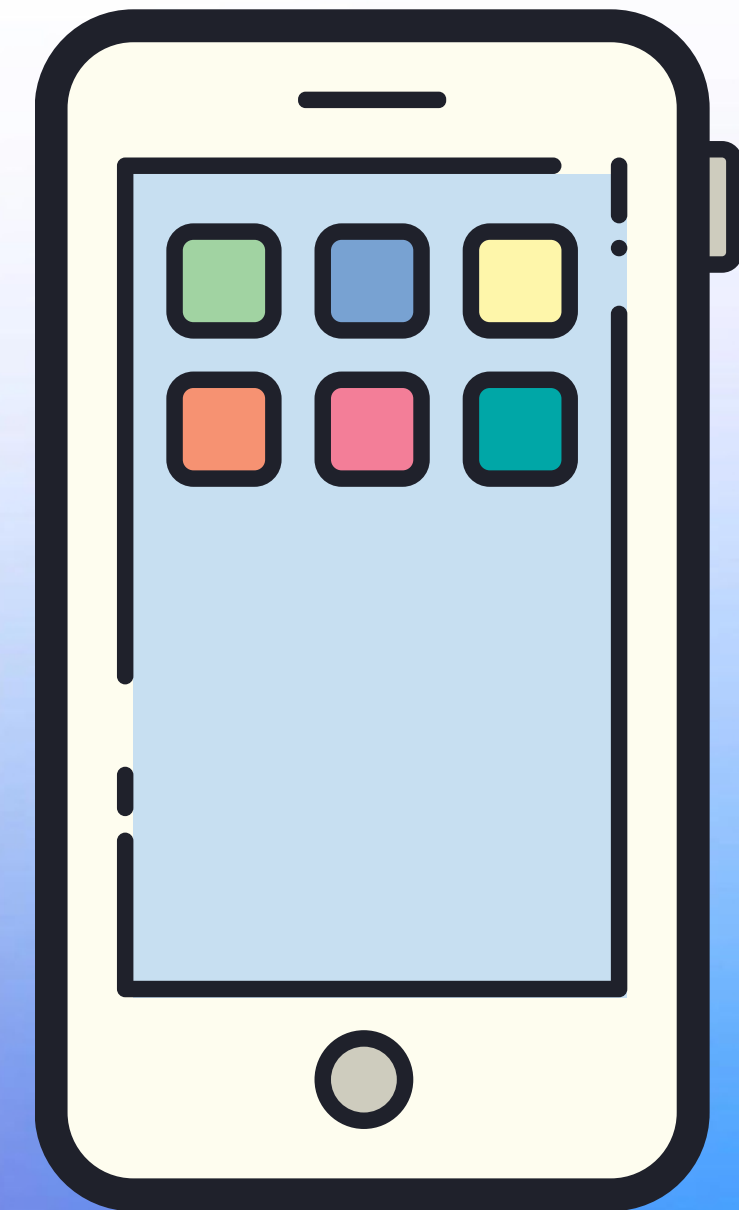
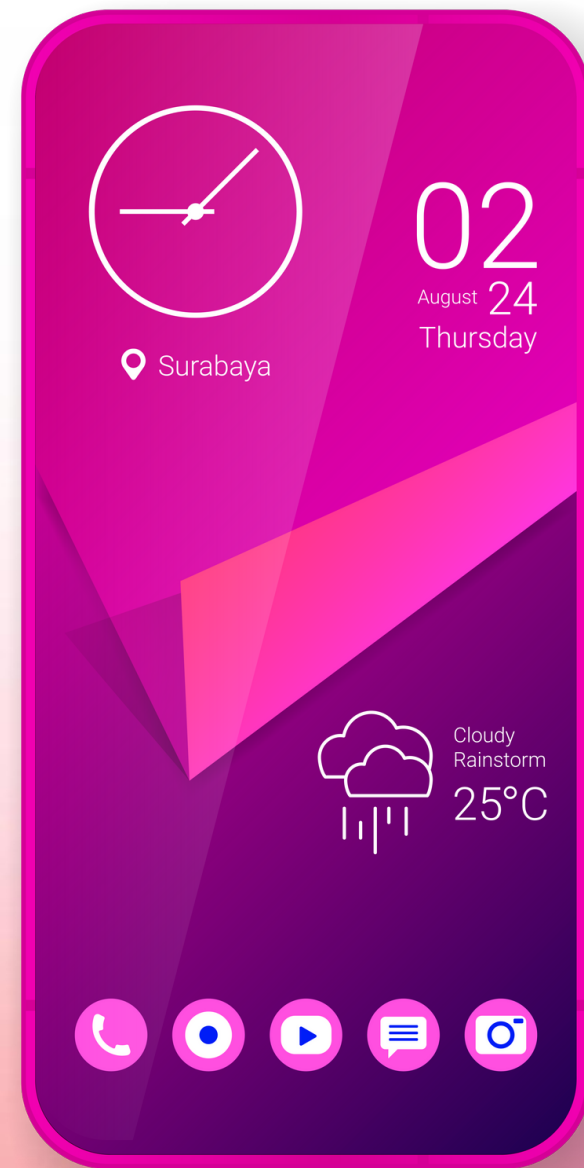


Turn off Animations on low data mode or low memory

MYTH BUSTER

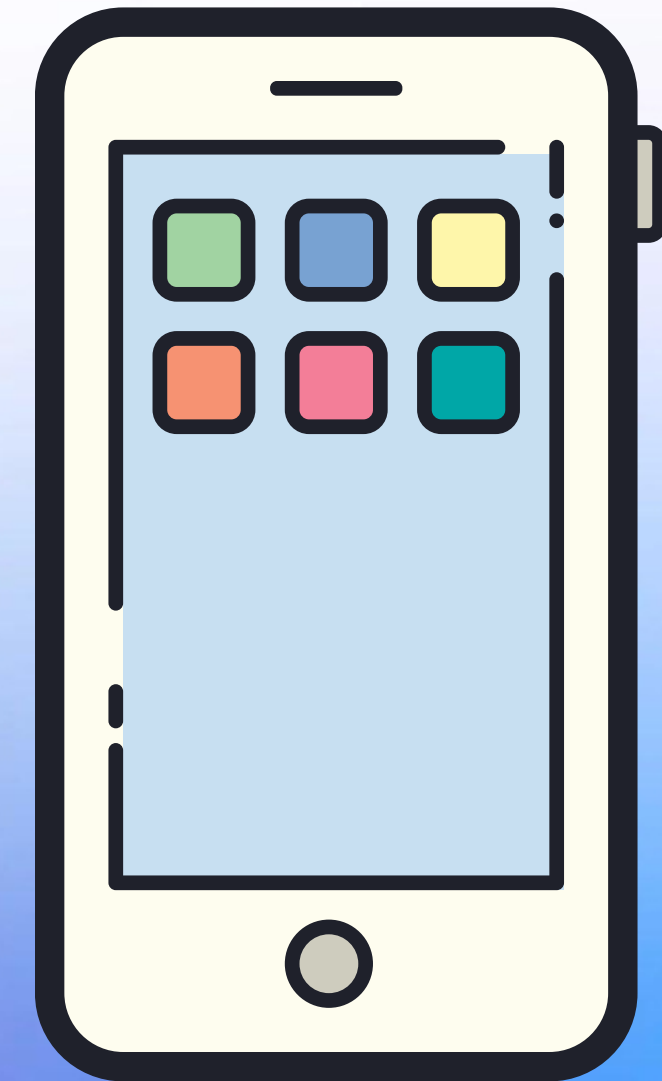
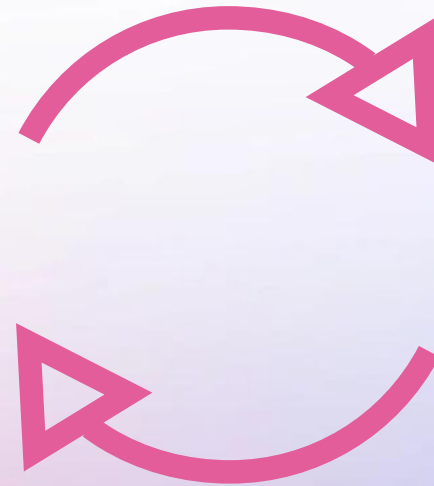
Testing app performance on an iPhone is enough...

Not everyone has an high-end iPhone

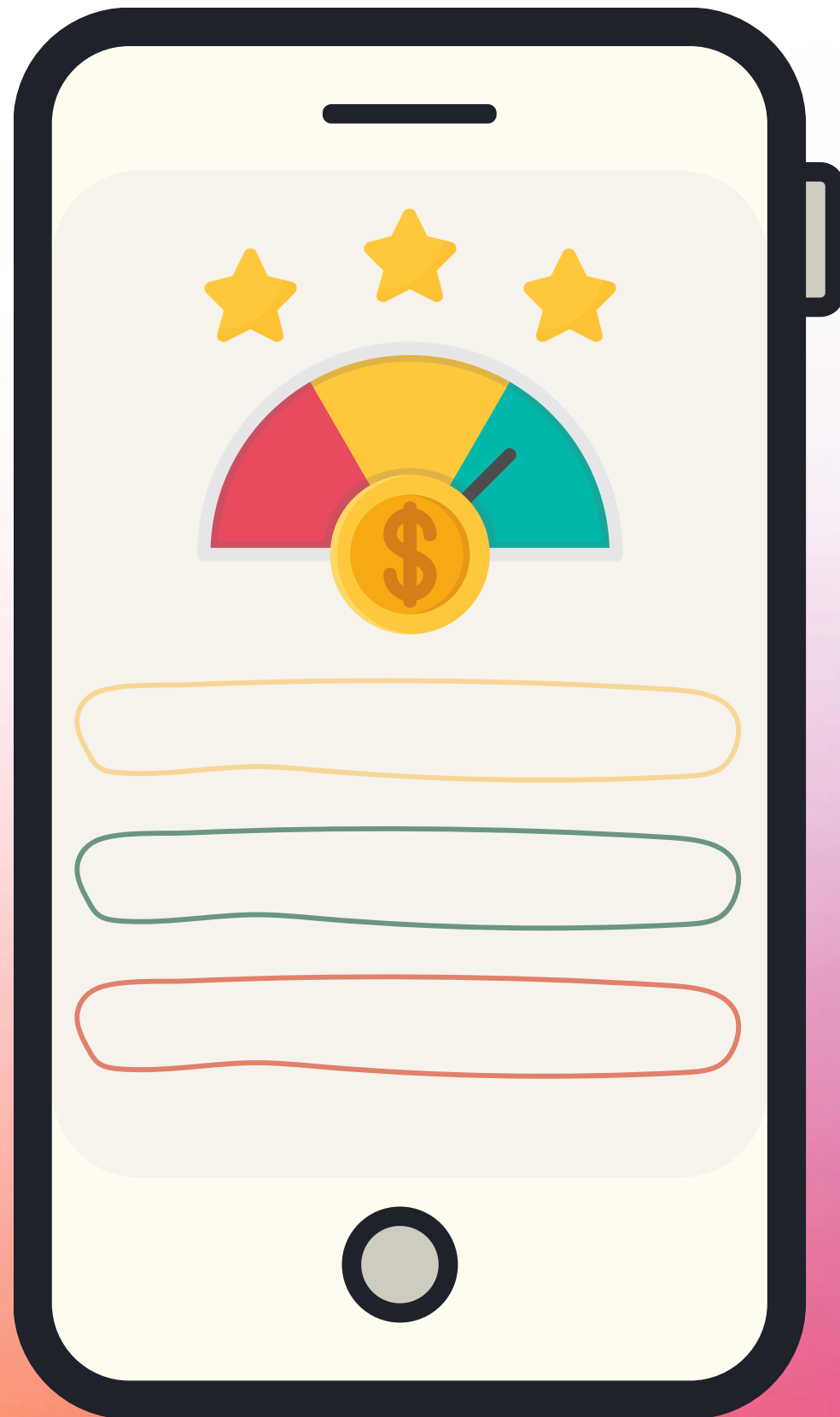


How to actually test Performance?

**Test on
Lower-end
Devices**



Let's Measure Performance



React Dev Tools using Profiler

Disable Dev Mode and test in
Prod Mode

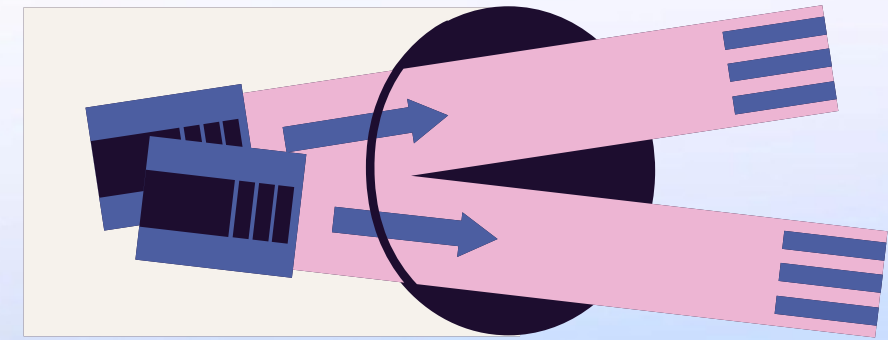
When should you go native vs React Native?



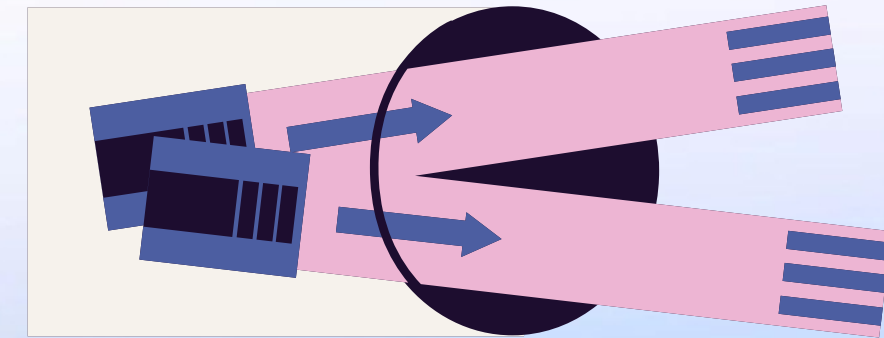
send strip



take picture



**Scan photo
again**



**Error, retake
photo**

Quick Announcement

**How many of us have to jump
in the deep end and start
leading**

developertoleader.com

[ABOUT THE INSTRUCTOR](#)[WHO IS THIS FOR](#)[CURRICULUM](#)[BONUS](#)[COMMUNITY](#)[PRICING](#)[FAQ](#)[🌟 Join Now](#)

Become a Successful Engineering Leader

Learn how to lead a team successfully even if you've never managed or led anyone before.



Taught 10,000+ Happy Students online

[🌟 JOIN NOW](#)

DOORS TO DEVELOPER TO LEADER CLOSE IN...

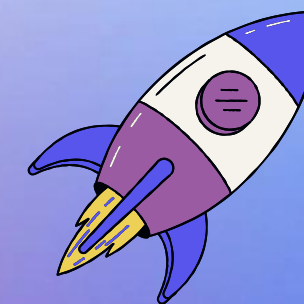
DEVELOPER TO LEADER PROGRAM



A program to help developers transition into
Engineering Leadership

Closing doors soon

DEVELOPERTOLEADER.COM



**DEVELOPER
TO LEADER**

@kulkarniankita9

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

FEATURE DEVELOPMENT

Make performance part of your practice in user stories and testing

Thank you!

Check out developertoleader.com

All my slides are at bit.ly/appjs23



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

Let us build a Dark Mode App



Dark / Light

Toggle dark mode



Dark / Light

Toggle dark mode

Let's review an app

Logo



Toggle Button Handler

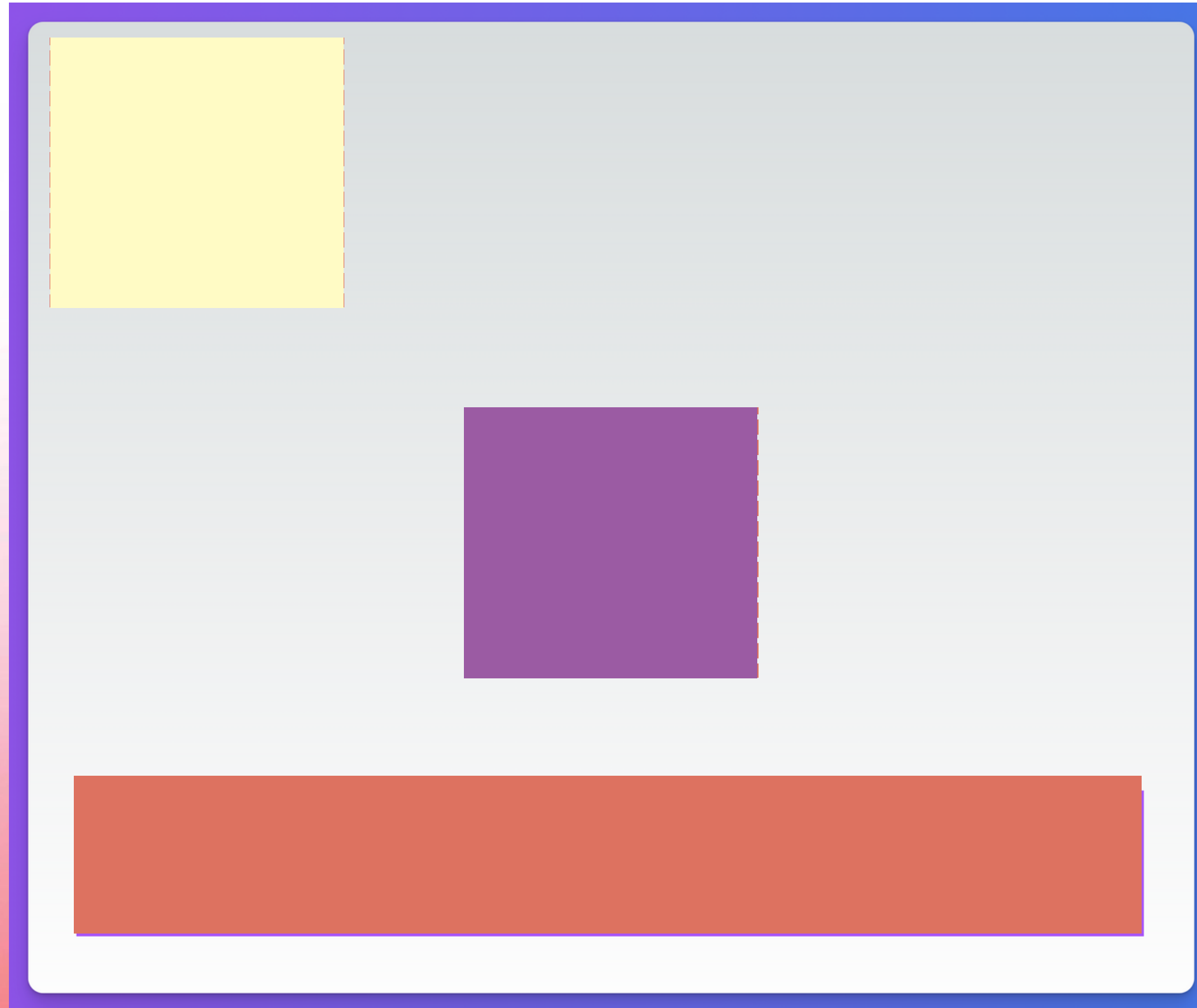
Toggle Dark Mode

Dark mode is disabled.

Footer

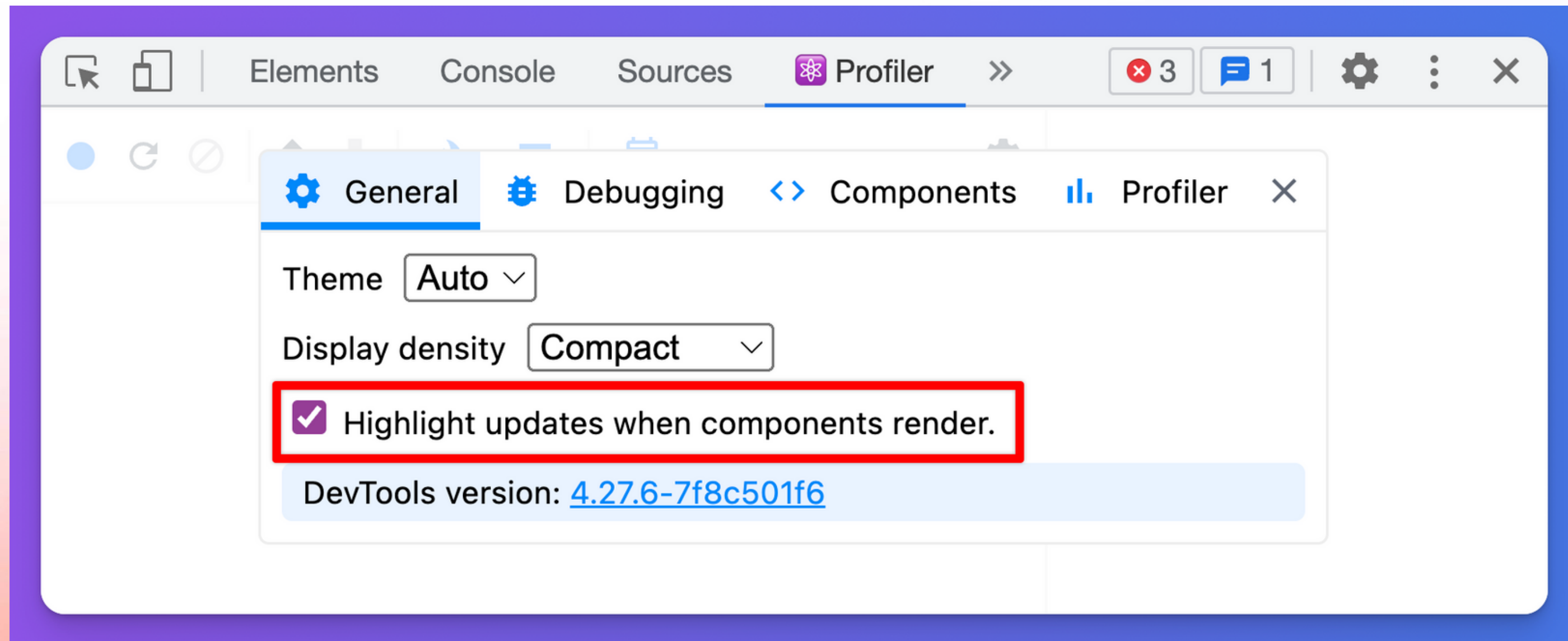
Anky Coby Bean Inc.

React captures a Snapshot of this UI



HOT TIP

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';

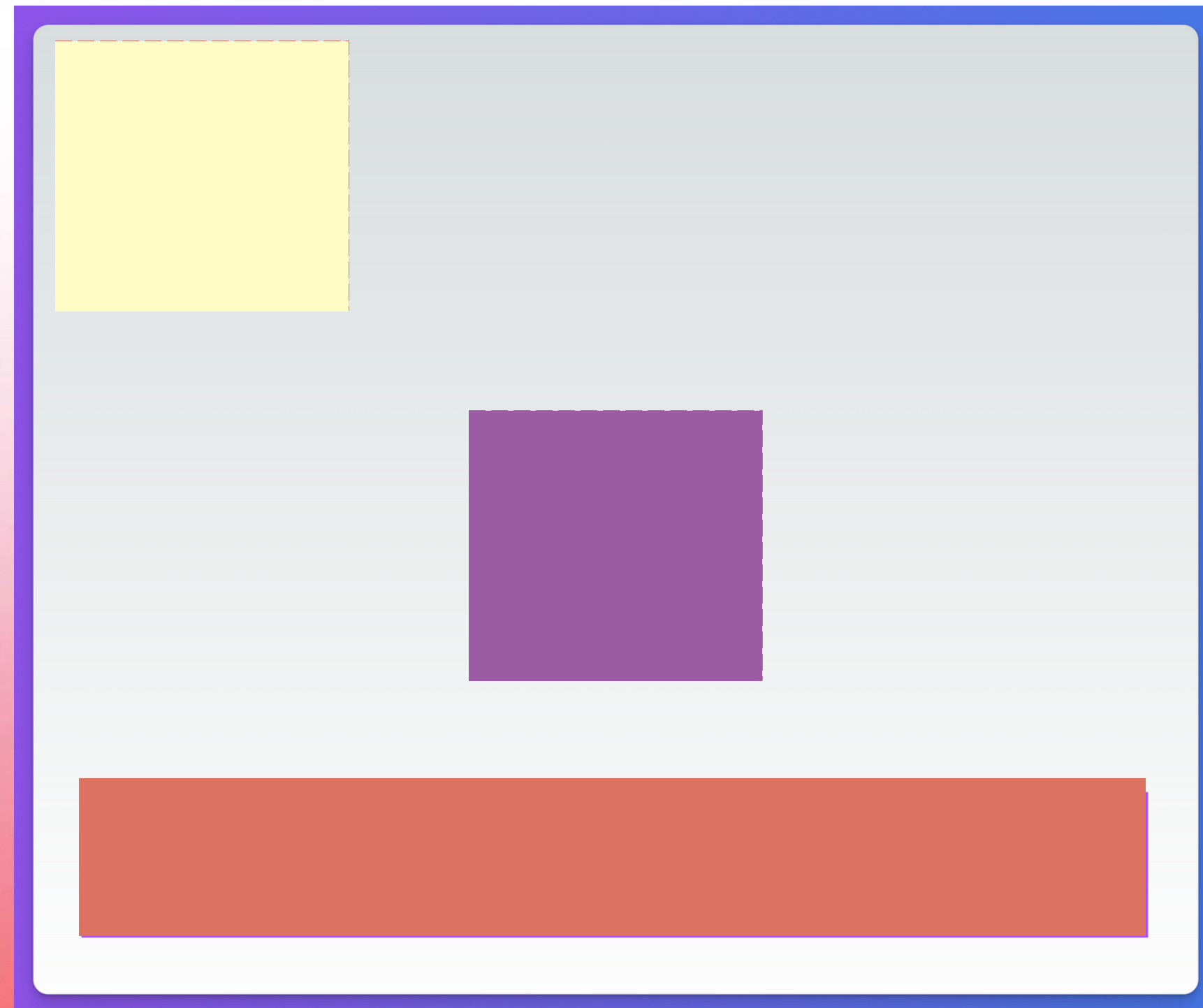
export default function Home() {
  const [darkMode, setDarkMode] = useState(false);

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

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

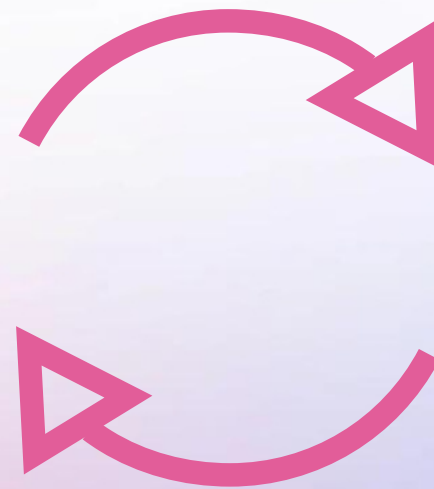
      <Footer />
    </main>
  );
}
```


**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**

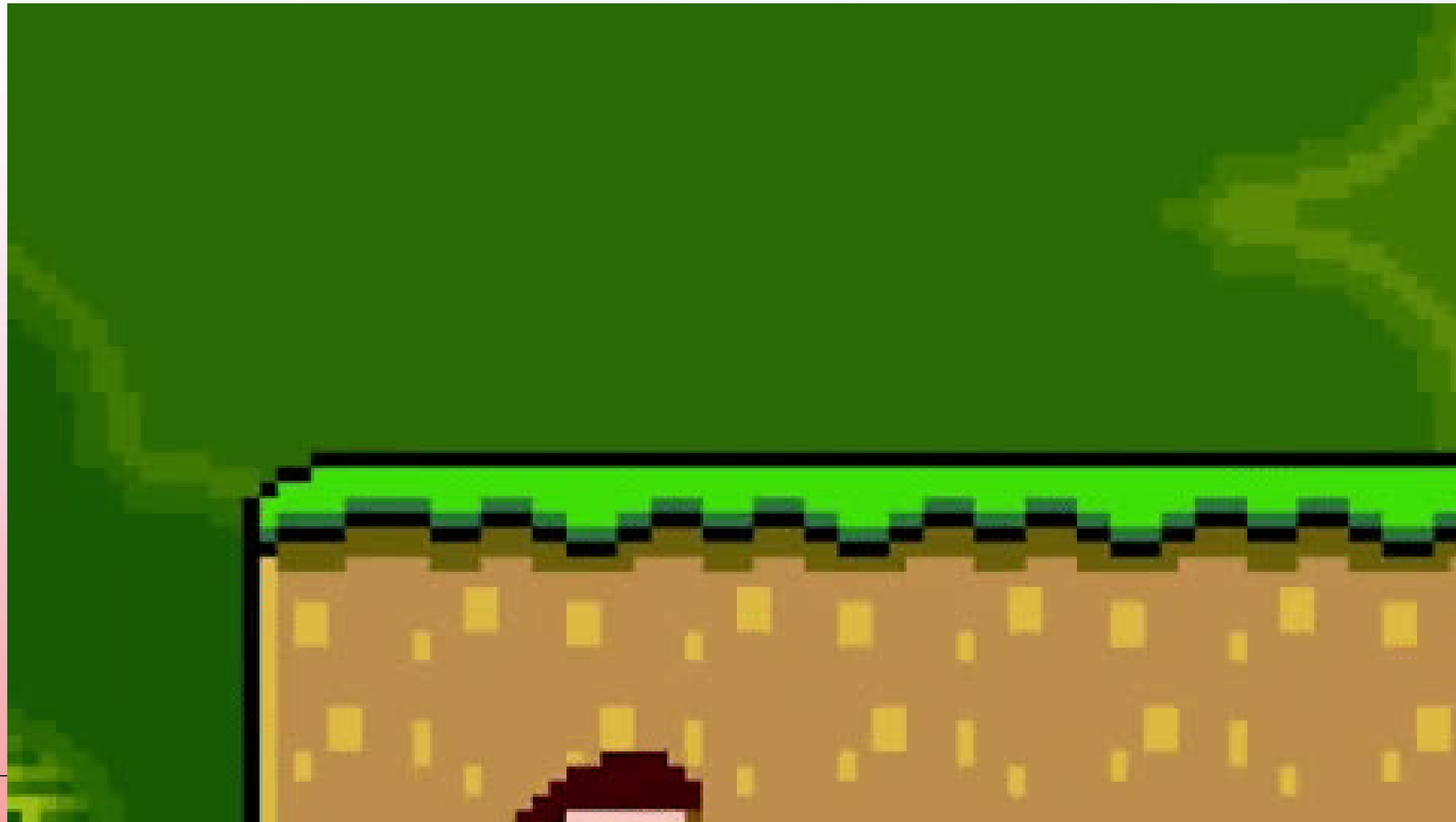


**REDUCE THE
NUMBER OF
RE-RENDERS**

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?

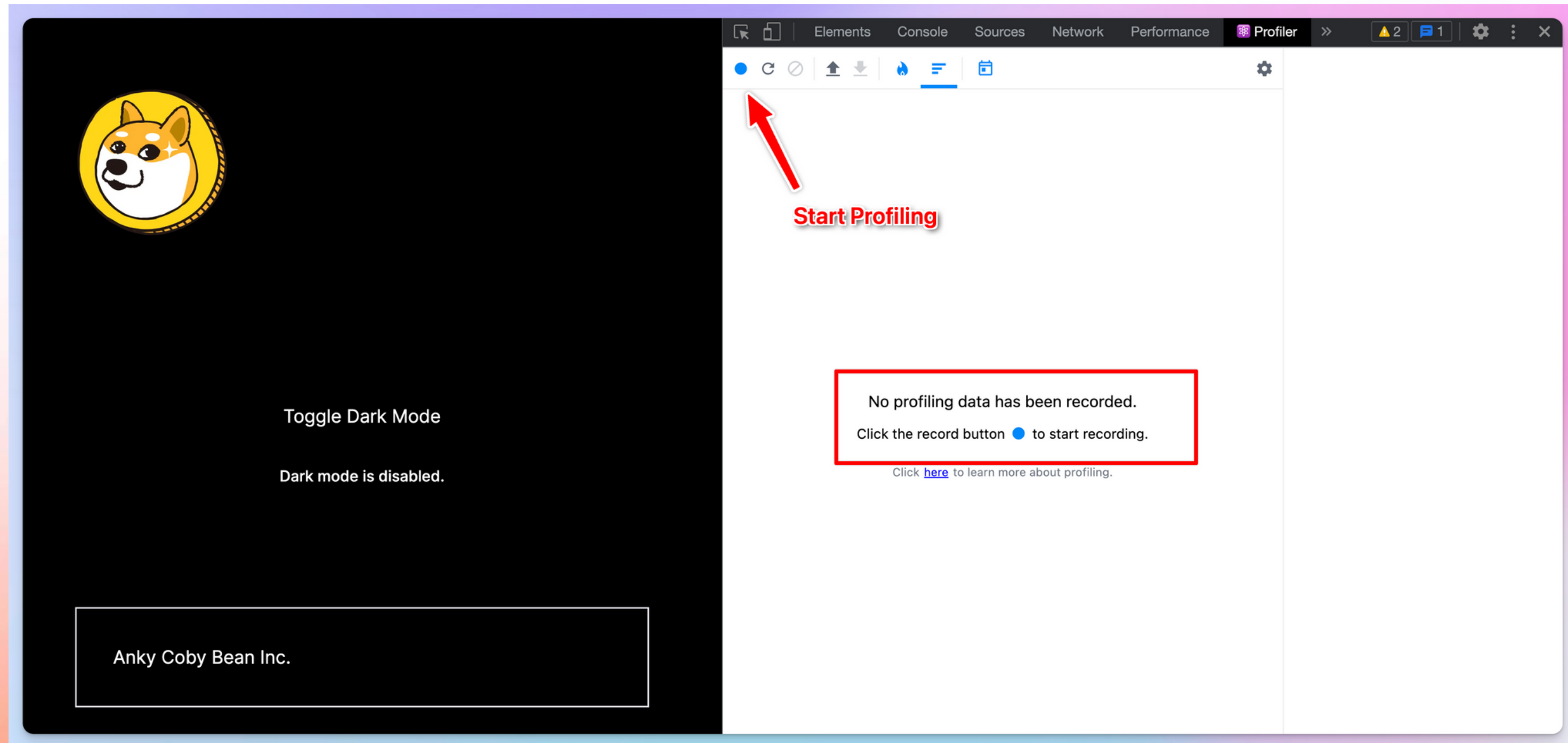
How does React work?



React's UI + React State



Let's Profile our lil app



Let's Profile our lil app

Clicking Logo

The image shows a web application interface on the left and the React DevTools Profiler on the right. The application has a dark theme with a logo in the top left, a 'Toggle Dark Mode' button, and a footer 'Anky Coby Bean Inc.'. The Profiler shows a commit triggered by clicking the logo. The commit information panel on the right indicates the update was committed at 1.8s with a render duration of 1.3ms, caused by the 'Home' component. The component tree on the left shows the 'Logo' component rendering in less than 0.1ms of a total 0.8ms commit time, with the reason being 'The parent component rendered.'

Logo | 756px × 191px

Toggle Dark Mode

Dark mode is enabled.

Anky Coby Bean Inc.

Elements Console Sources Network Performance Profiler

Anonymous (ForwardRef) (0.4ms)

Anonymous (ForwardRef) (0.2ms)

Footer (0.1ms)

Home (0.1ms)

Logo

<0.1ms of 0.8ms

Why did this render?

The parent component rendered.

Commit information

Priority: Immediate

Committed at: 1.8s

Render duration: 1.3ms

What caused this update?

Home

How do we solve for this?

Let's wrap Logo & Footer in memo

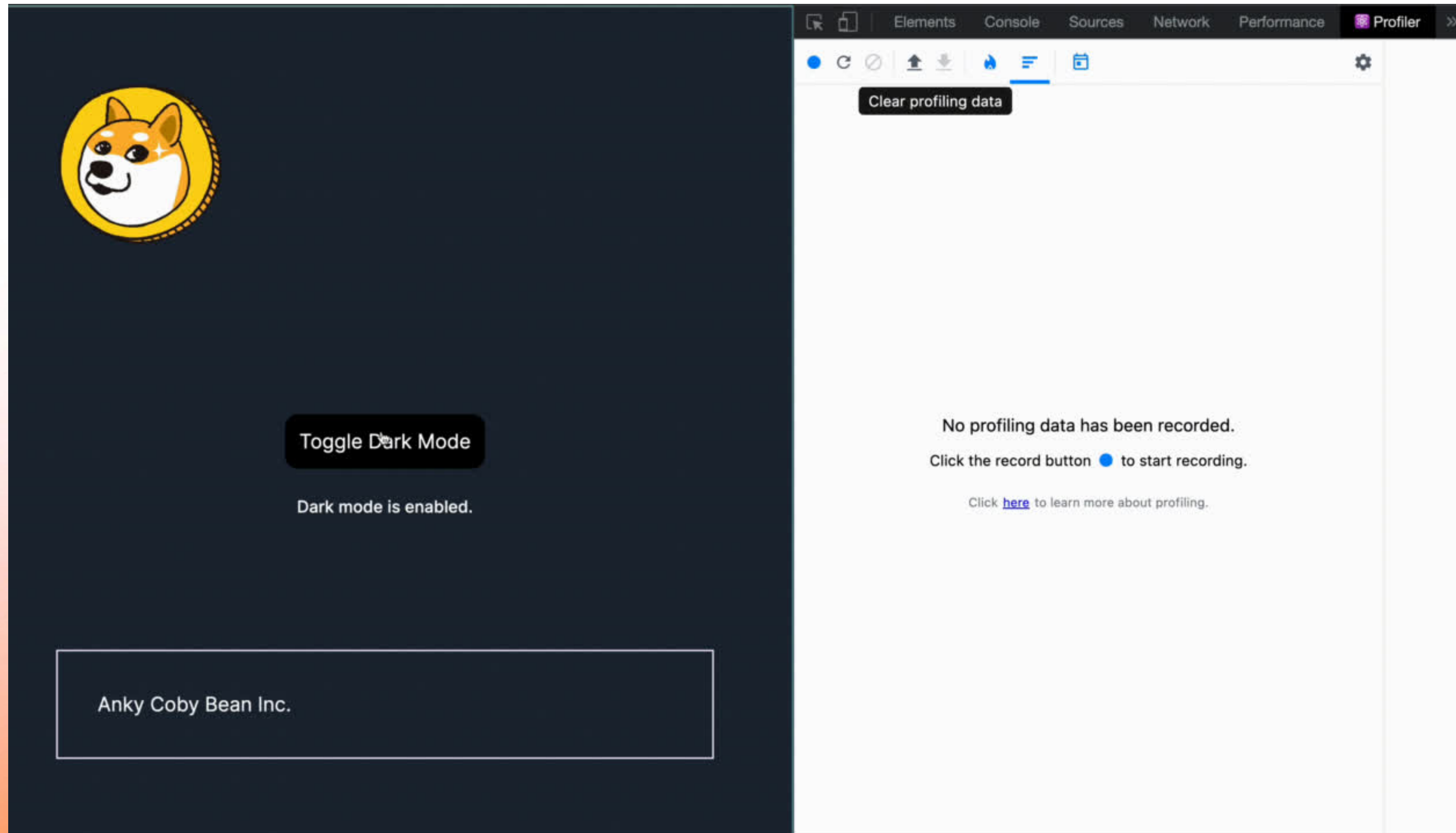
```
import Image from 'next/image';
import React, { memo } from 'react';

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



The image shows a web application interface on the left and a browser's developer tools Profiler on the right.

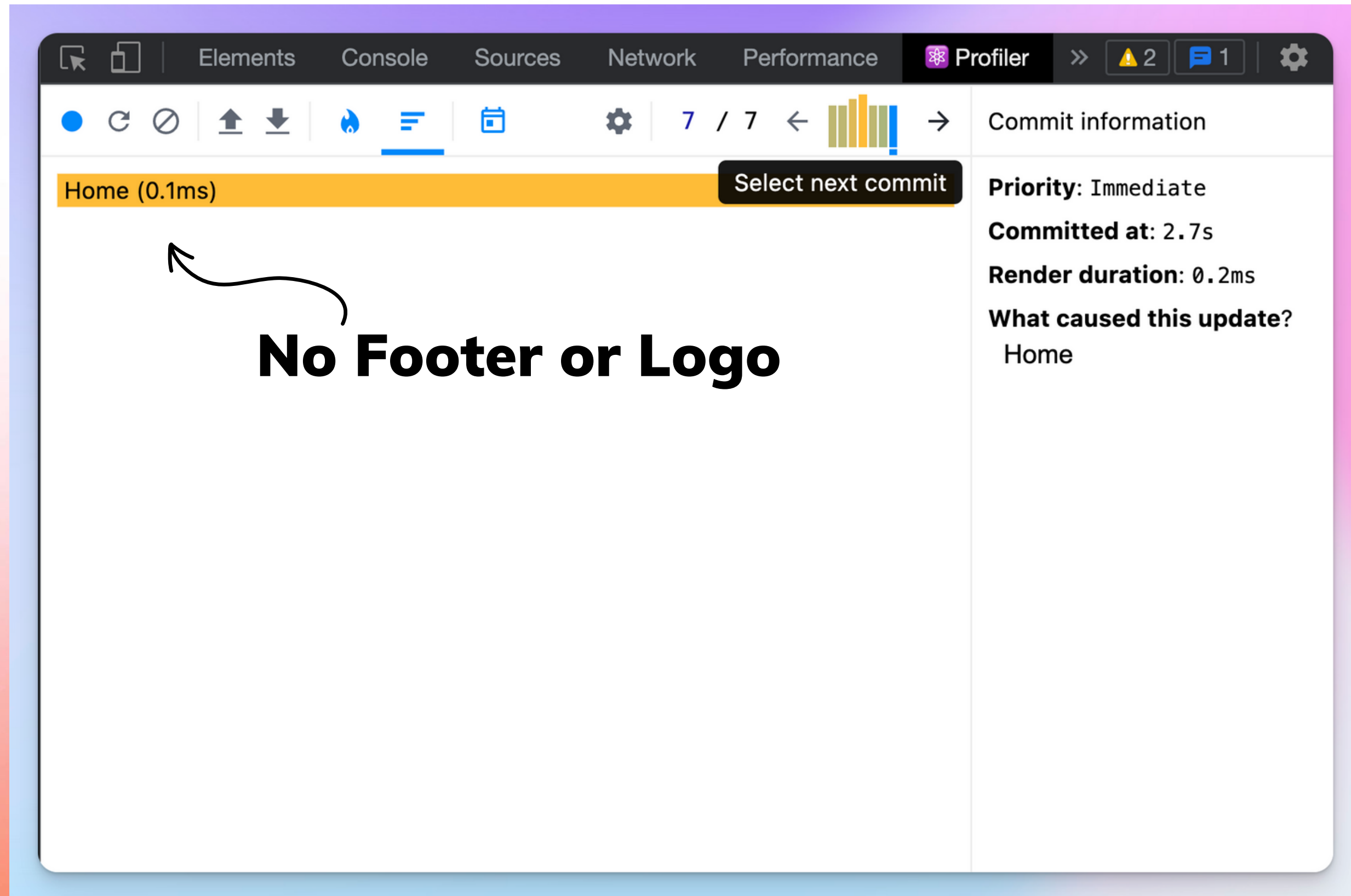
Web Application Interface:

- Top left: A circular logo featuring a Shiba Inu dog's face.
- Center: A button labeled "Toggle Dark Mode".
- Below the button: The text "Dark mode is enabled."
- Bottom: A white rectangular box containing the text "Anky Coby Bean Inc."

Browser Developer Tools (Profiler):

- Top bar: Includes tabs for Elements, Console, Sources, Network, Performance, and Profiler (which is active).
- Toolbar: Contains icons for recording, pausing, and other profiling actions.
- Content area: Displays the message "No profiling data has been recorded." and "Click the record button ● to start recording.".
- Footer: A link that says "Click [here](#) to learn more about profiling."

If we pause it



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



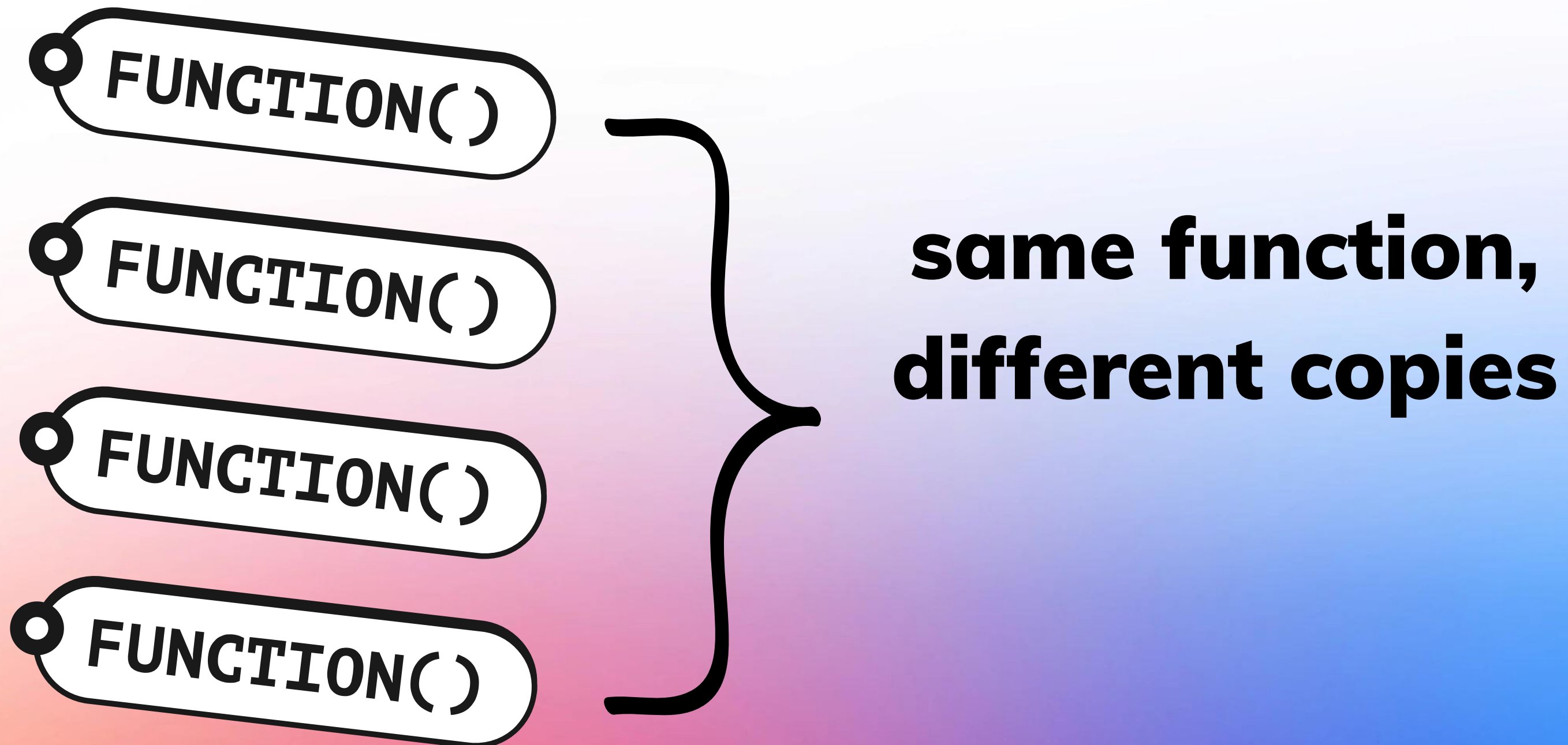
**RETURNS
CACHED
FUNCTION**

**[]
DEPENDENCY
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 Example

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 < 1000000000; 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