SOME THOUGHTS AND VIBES ON

Optimizing React Applications

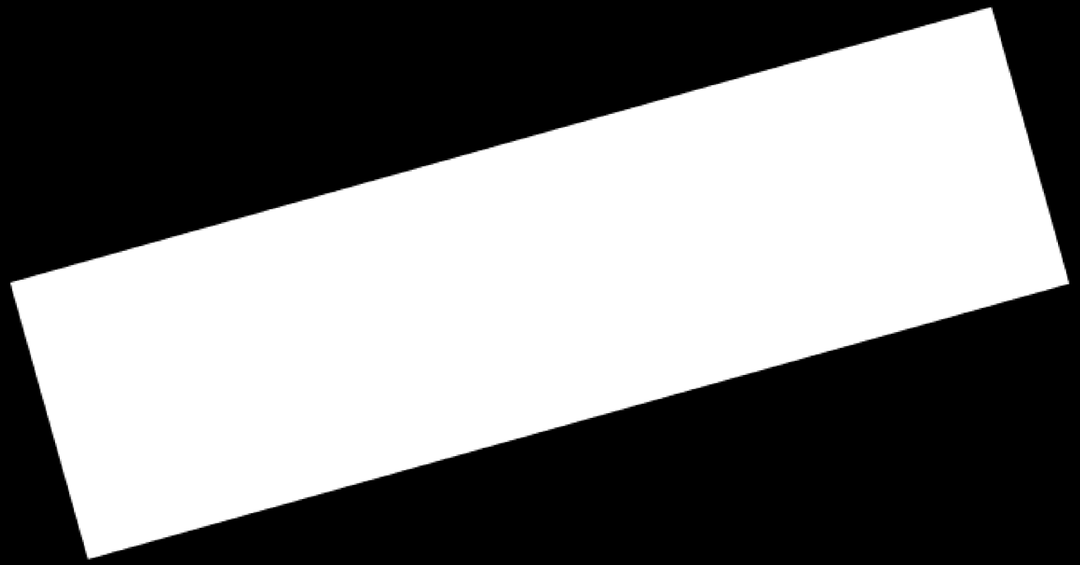
@keerthanak17

I’m Keerthana Krishnan (KK)

From Cochin, India living in Munich, Germany

Software Engineer
Yamdu

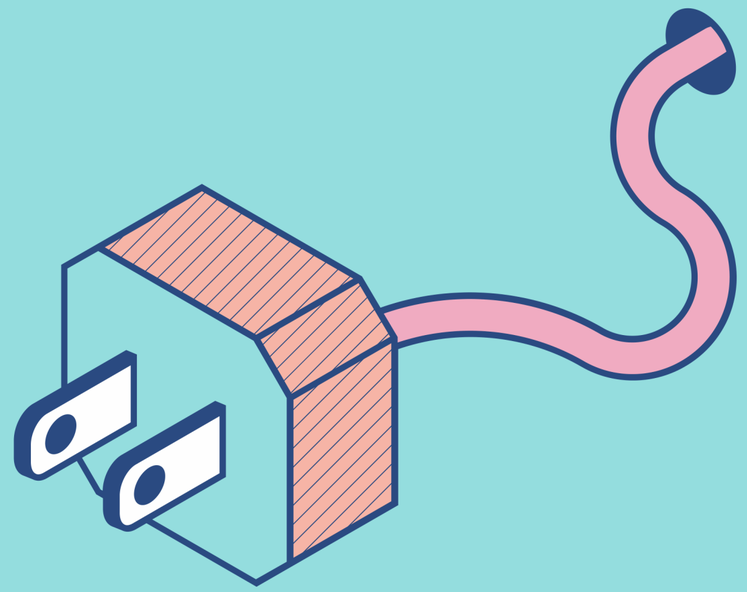
@keerthanak17

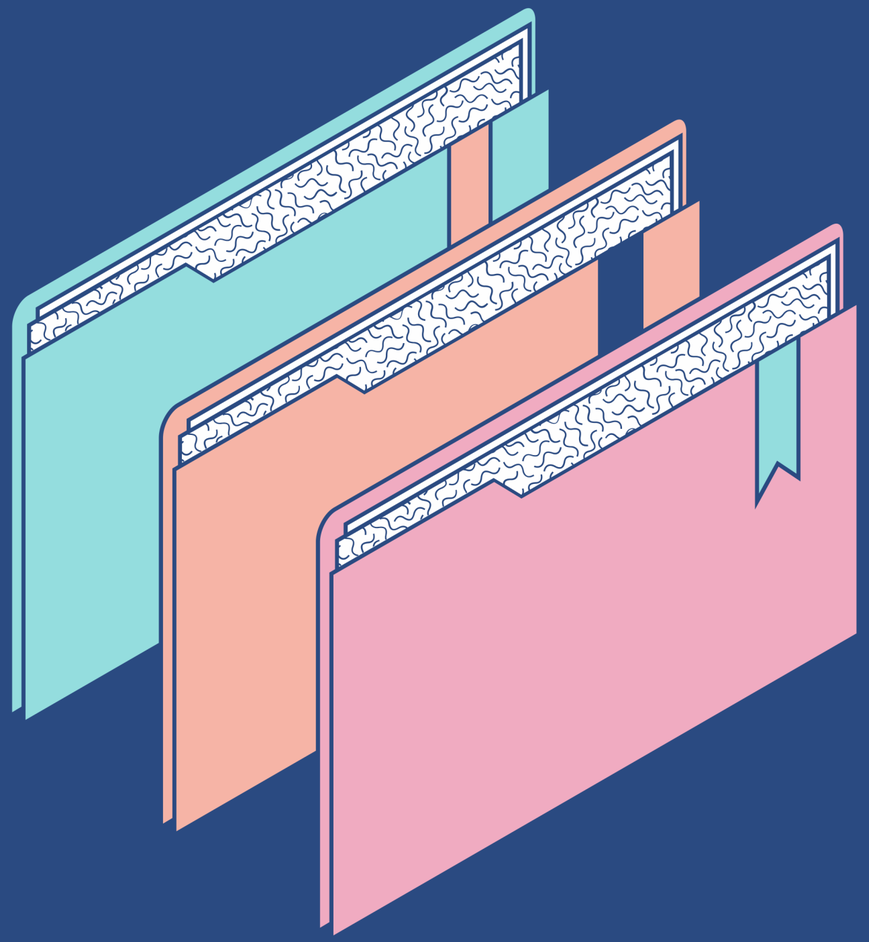


Yamdu®

Some caveats before we start

- React specific examples, but these points work for any JavaScript framework
- Front-end focused, with no back-end/ infra changes expected
- Principles, not silver bullets





Agenda

TALKING ABOUT REACT APPLICATION PERFORMANCE OPTIMIZATION

- What are the metrics of React performance measurement?
- Why is this topic important?
- How can we improve the performance of your web application?

WEB CORE VITALS

AN IN-DEPTH GUIDE TO MEASURING CORE WEB VITALS

“ ... set of three metrics designed to measure the “core” experience of whether a website feels fast or slow to the users, and so gives a good experience”

- Barry Pollard

@keerthanak17

Performance - Interactivity - Visual stability

CORE WEB VITALS

(Loading)

LCP

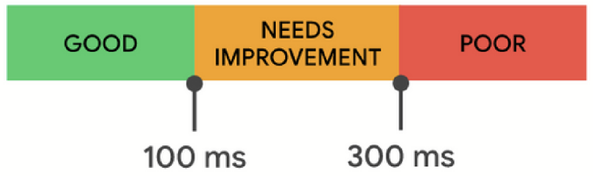
Largest Contentful Paint

To provide a good user experience, LCP should occur within 2.5 seconds of when the page first starts loading

(Interactivity)

FID

First Input Delay



To provide a good user experience, pages should have a FID of 100 milliseconds or less

(Visual Stability)

CLS

Cumulative Layout Shift

To provide a good user experience, pages should maintain a CLS of 0.1. or less

@keerthanak17

Vs “lab-based” web performance tools (Lighthouse)

WHY THESE METRICS ARE THE NEW STANDARD

Basis

Google uses anonymized data, field metrics or Real User Metrics (RUM) from Chrome users

---

Distinction

Lab-based tools run page loads on simulated networks and devices and then tell you what the metrics were for that test run. If you run Lighthouse on your high-powered developer machine and get great scores, that may not be reflective of what the users experience in the real world

---

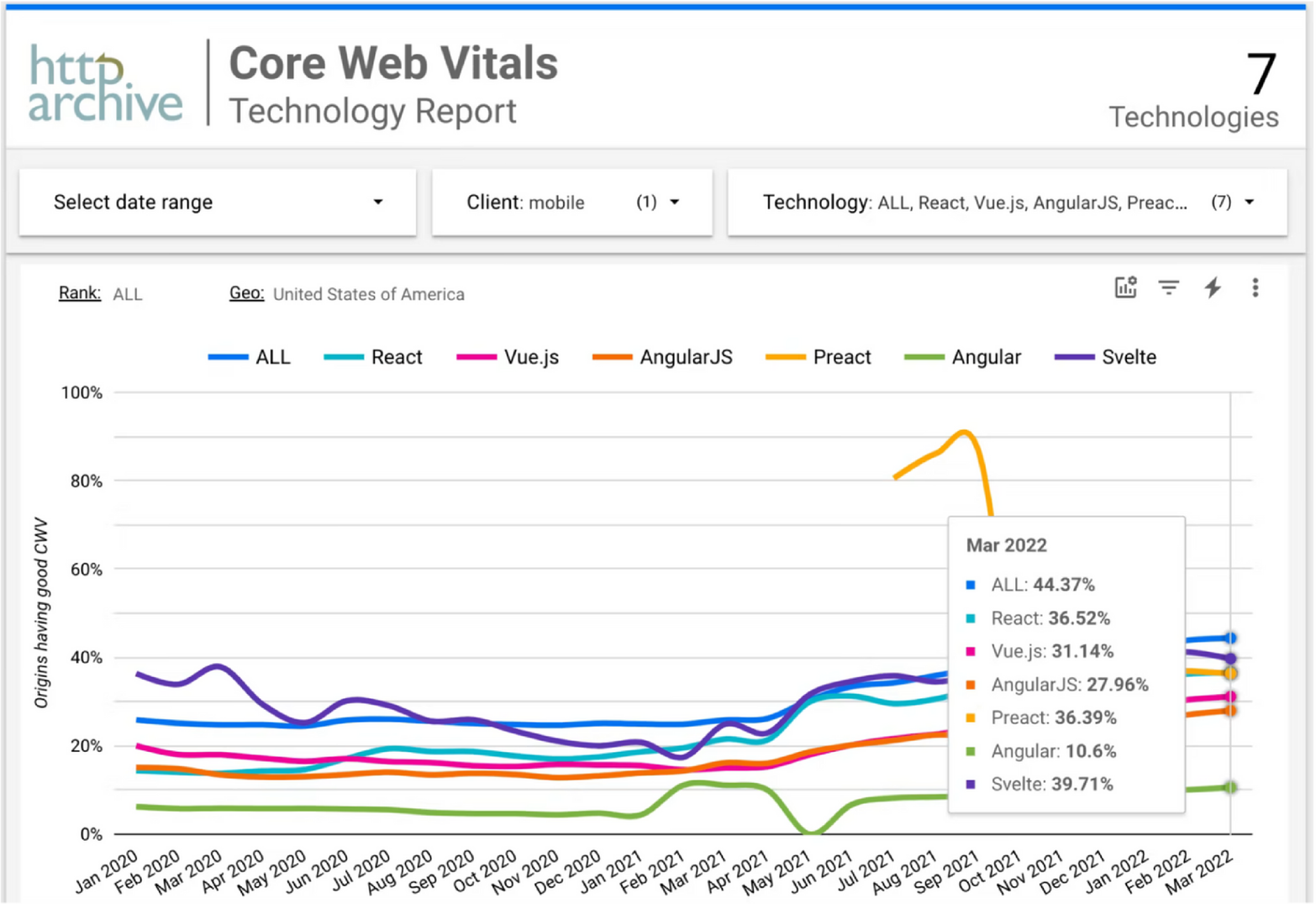
Importance

Google considers “Page Experience” as part of Search ranking

Percentage of websites with all green CWV for leading frameworks, sessions on mobile in the USA

[HOW TO USE GOOGLE CRUX TO ANALYZE AND COMPARE THE PERFORMANCE OF JS FRAMEWORKS]

- Dan Shappir

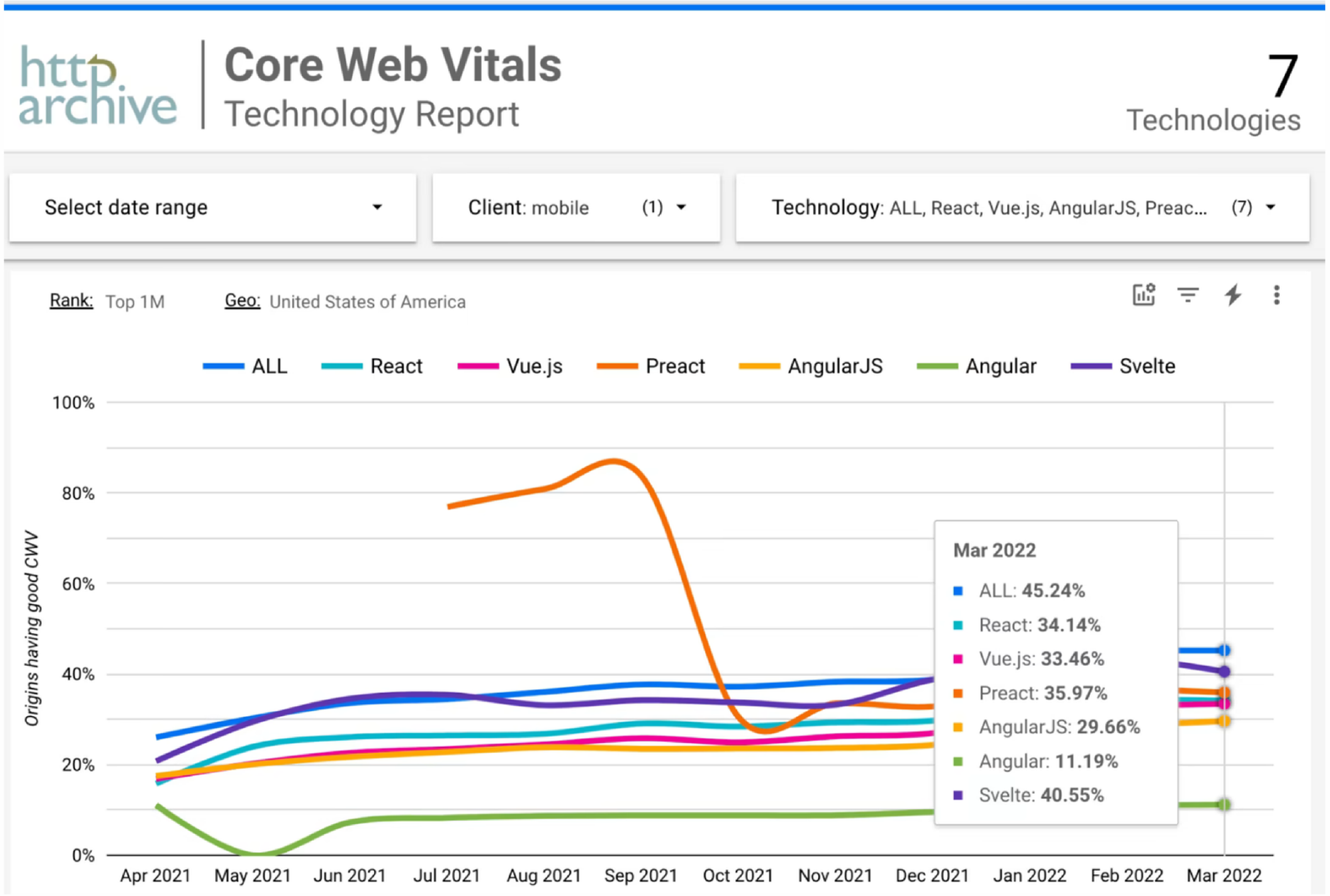


@keerthanak17

Percentage of websites with all green CWV for leading frameworks, sessions on mobile in the USA for the top 1M websites

[HOW TO USE GOOGLE CRUX TO ANALYZE AND COMPARE THE PERFORMANCE OF JS FRAMEWORKS]

- Dan Shappir



@keerthanak17

Presenting the conclusions from analyzing the Core Web Vitals report

HOW TO USE GOOGLE CRUX TO ANALYZE AND COMPARE THE PERFORMANCE OF JS FRAMEWORKS

- Dan Shappir

LCP scores are a strong match for CWV as a whole but when we limit to the top 1,000,000 sites React doesn't improve so, Vue catches up with it

For Vue, both the ratio of good LCP and CLS improve when we check top sites only. For React, on the other hand, LCP remains mostly the same, while CLS actually degrades.

FID is essentially meaningless, with all frameworks achieving a nearly perfect score

When limiting for 1M websites

HOW TO USE GOOGLE CRUX TO ANALYZE AND COMPARE THE PERFORMANCE OF JS FRAMEWORKS

- Dan Shappir

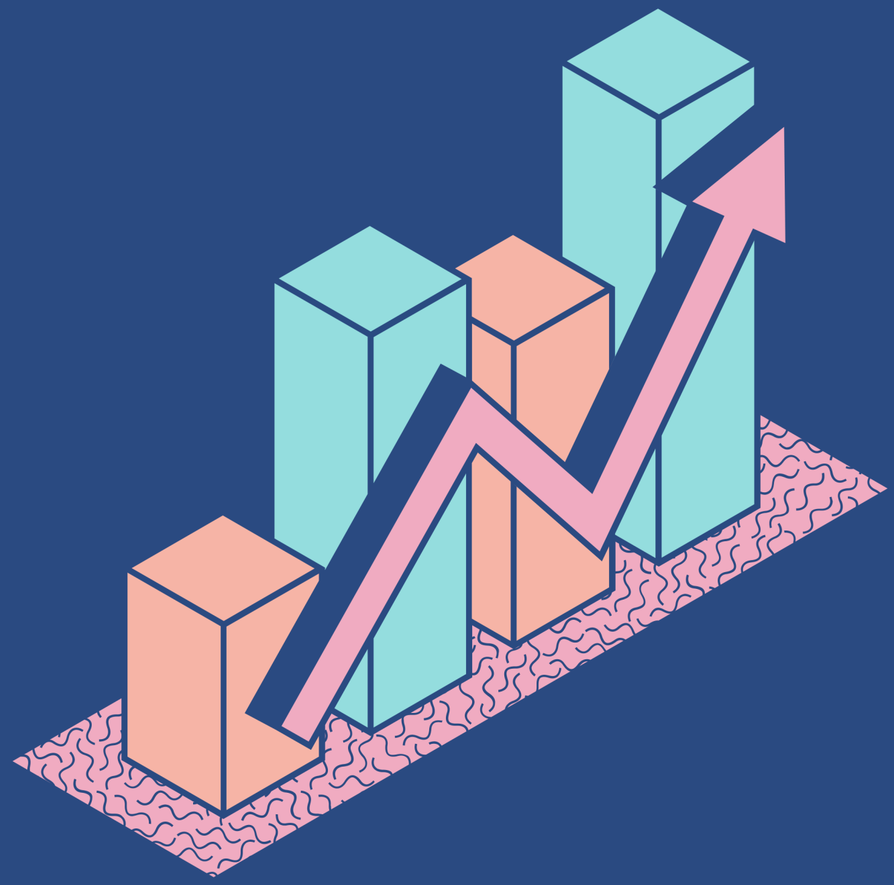
The amount of JavaScript downloaded increases, with the interesting exception of Vue.

Whatever greater expertise the developers working on top sites may have is perhaps cancelled due to the extra functionality provided by such websites

These sites download significantly less image data by leveraging techniques like lazy loading and newer image formats.

“React sites actually go down when limiting the number of sites measured by traffic”

WHY IS IT THAT REACT DEVELOPERS WITH GREATER EXPERTISE ARE APPARENTLY LESS LIKELY TO PRODUCE FAST LOADING WEBSITES?



@keerthanak17

About FID

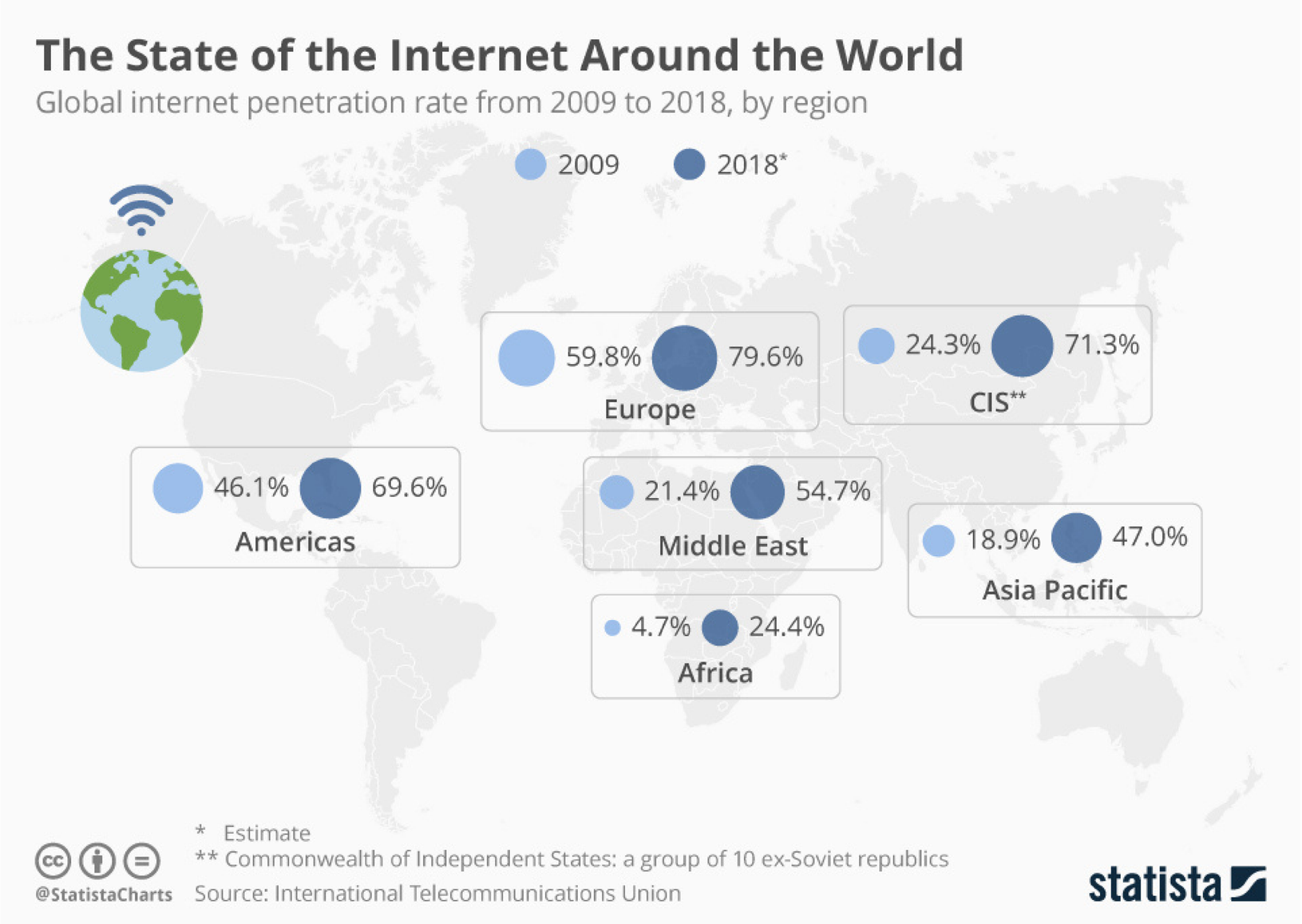
“ ... THE FID METRIC IS ESSENTIALLY MEANINGLESS, WITH ALL FRAMEWORKS ACHIEVING A NEARLY PERFECT SCORE ”

“... INP takes all interactions into account, reporting one of the slowest over the entire lifetime of the page. And, rather than only measuring the delay portion, INP measures the full duration from the start of the interaction ”

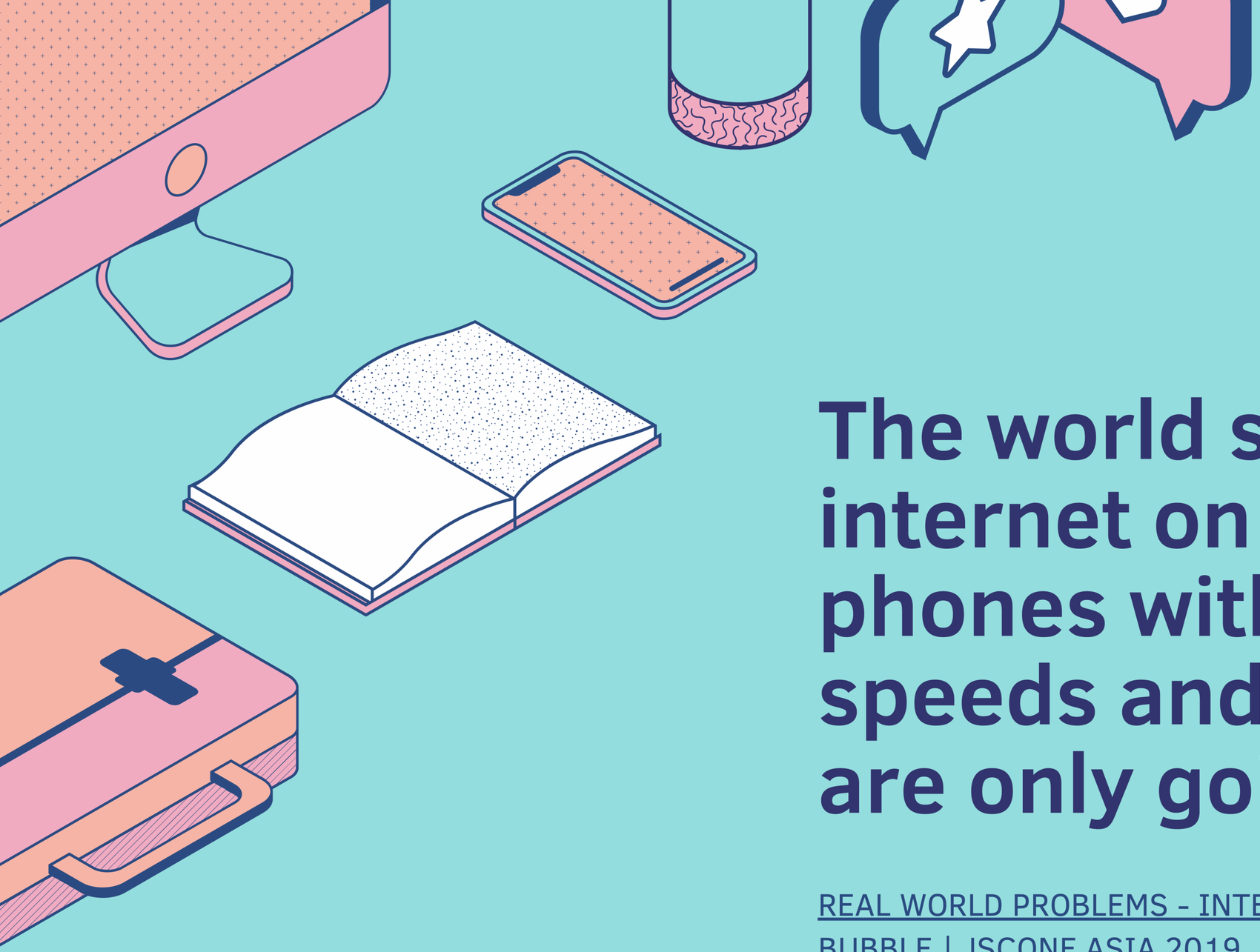
Internet usage around the world

Mobile phones far outpace any other device connected to the internet

REFERENCE



@keerthanak17

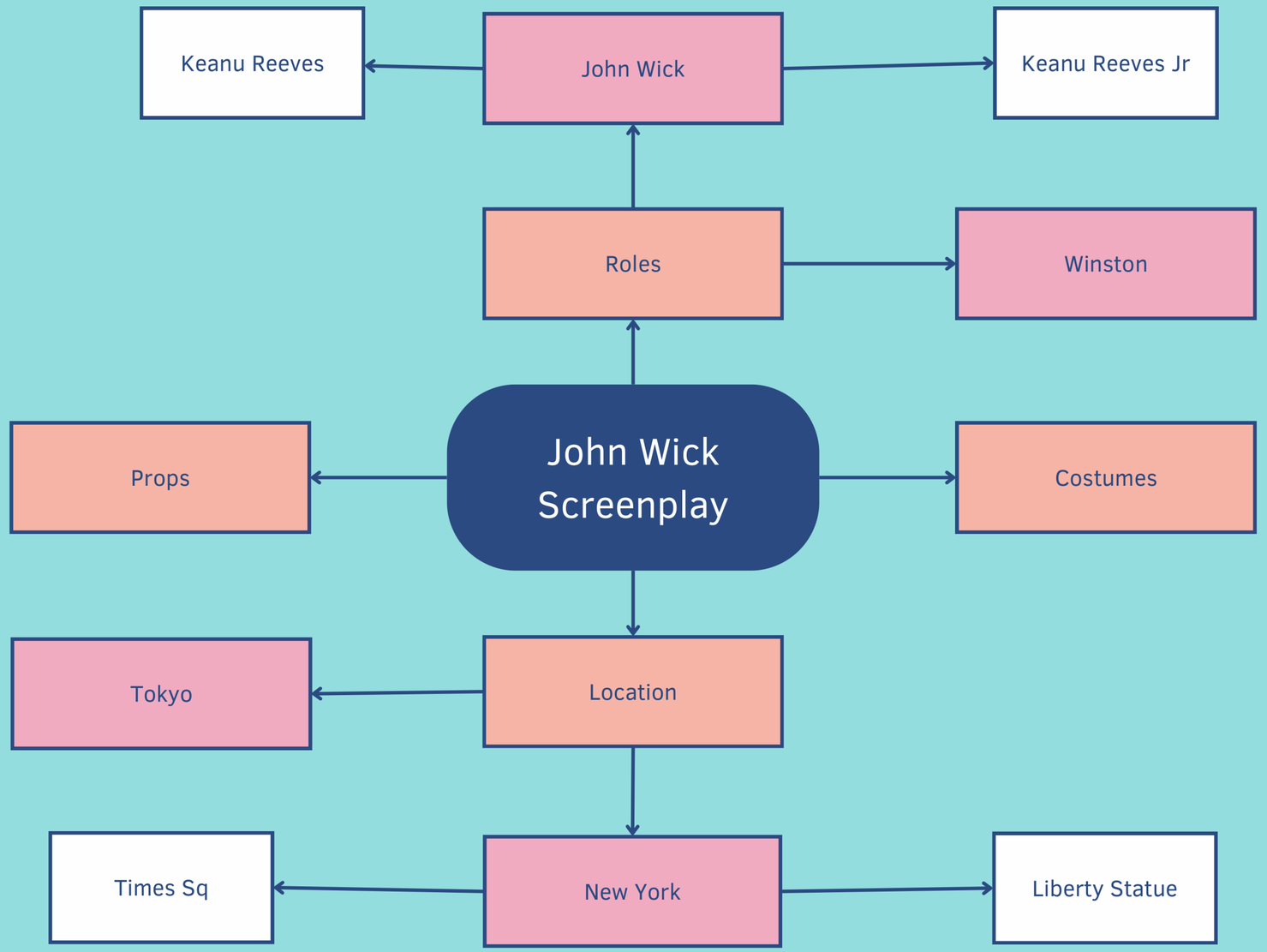


The world still uses the internet on crappy phones with slow speeds and the numbers are only going up

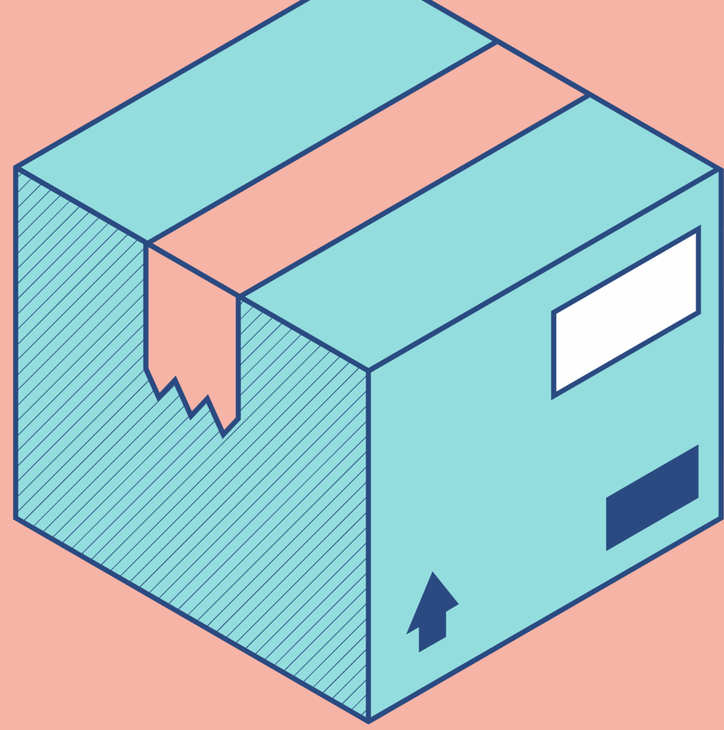
REAL WORLD PROBLEMS - INTERNET BEYOND THE 1ST WORLD BUBBLE | JSCONF.ASIA 2019

@keerthanak17

Example: Manage a script



Largest Contentful Paint (LCP).



THE LARGEST CONTENTFUL PAINT (LCP) METRIC REPORTS THE RENDER TIME OF THE LARGEST IMAGE OR TEXT BLOCK VISIBLE WITHIN THE VIEWPORT, RELATIVE TO WHEN THE PAGE FIRST STARTED LOADING. FOCUSES ON WHEN THE MAIN CONTENT IS PAINTED, AS OPPOSED TO FIRST CONTENT

OPTIMIZATION TIPS -

Eliminate resource load delay

Ensure that your LCP resource loads alongside the first resource is loaded by that page

Eliminate element render delay

What blocks the LCP element from rendering immediately after its resource finishes loading

Reduce resource load time

Reduce resource size, its travel distance and the contention for network bandwidth

Reduce time to first byte

Nothing can happen on the frontend until the backend delivers that first byte of content

@keerthanak17

LCP improvement isn’t react specific

1 STEP

Improve server response times

Using CDNs, SSR and other techniques to get quicker responses from the backend

2 STEP

Optimize media files

Getting the right image sizes for the user’s device, using the right formats and further optimizing them

3 STEP

Minifying your code

Using the production builds, minifying the JS and CSS files and utilizing tree shaking

4 STEP

Caching & lazy loading

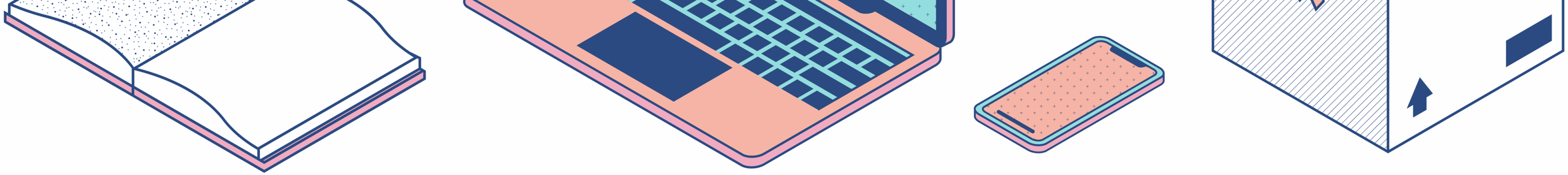
These allow for fast load times by reducing the amount of data transferred during the initial rendering process.

5 STEP

Eliminate render blocking JS & CSS

Performance metrics tools can flag render blocking resources. Deliver critical JS/CSS inline and defer others

@keerthanak17



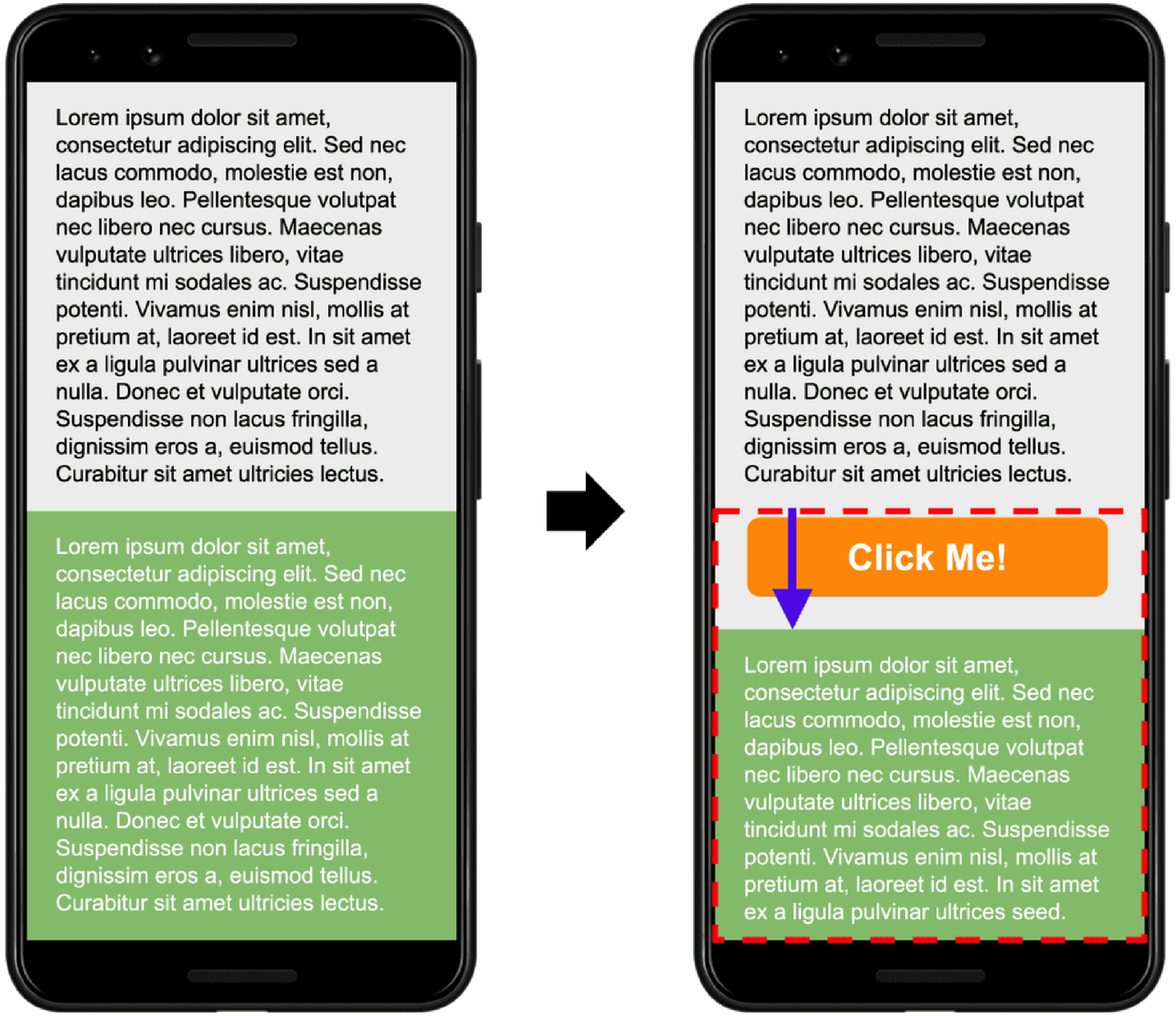
What’s counted in CLS?

- Measure of the largest burst of layout shift scores for every unexpected layout shift
- It’s a cumulative score counted for the entire lifespan of a page
- Layout shifts that occur in response to user interactions, (Ex: clicking a link) don’t count as long as the shift occurs close enough to the interaction

Whats’s in a layout shift?

- The browser looks at the viewport size and the movement of unstable elements in the viewport between two rendered frames
- Layout shift score is a product of the impact on the viewport area (Impact fraction) and the distance moved by the unstable element (Distance fraction)
- Ex: Dynamically added DOM elements

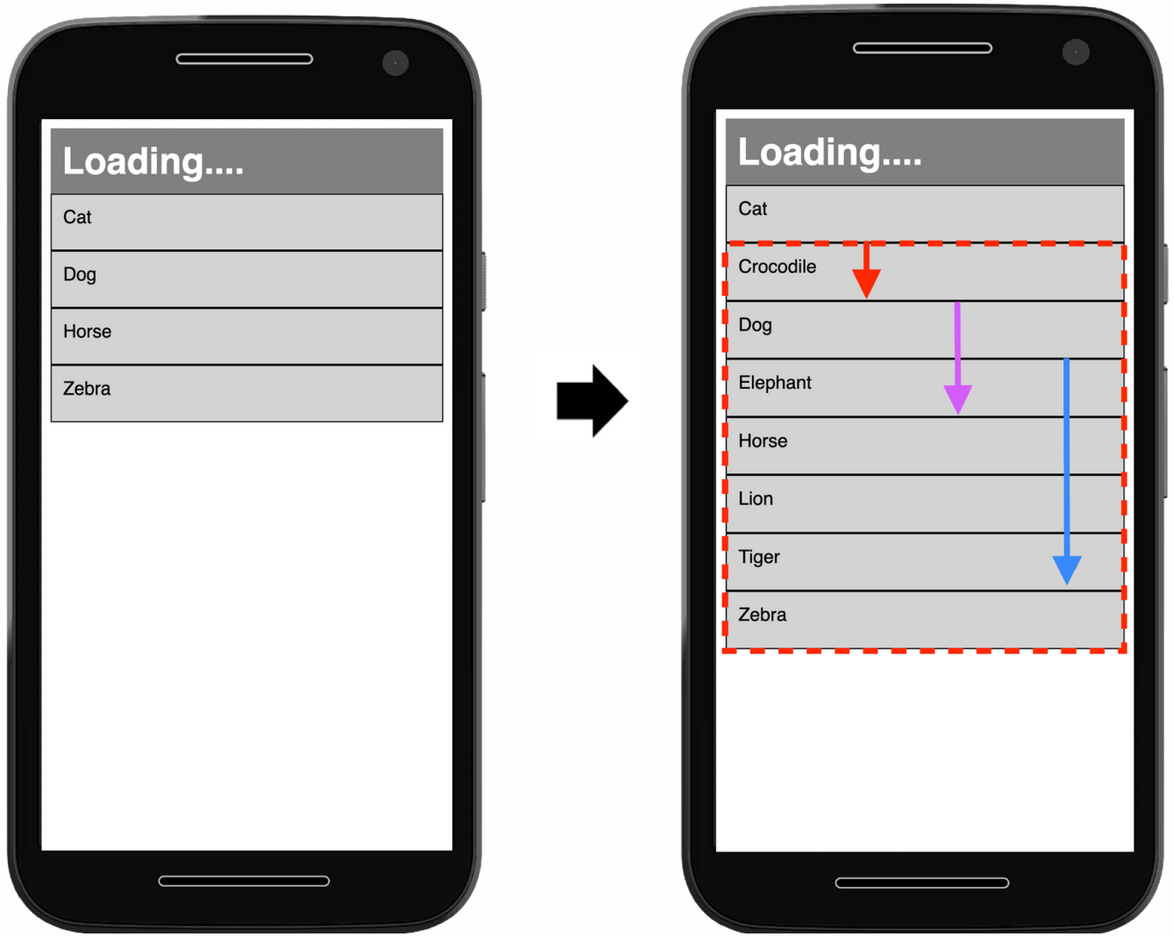
CLS Example with single unstable element



- The button is appended above the green box, pushing it down partially out of the viewport. The change of start position makes it unstable
- The gray box changes size, but its start position does not change so it's not an unstable element.
- The button was not previously in the DOM, so its start position doesn't change, its not unstable
- The layout score considers the fraction of the viewport impacted (50%, red dashes) and the fraction of the viewport distance that the element moved (14%, purple arrow)
- Layout shift score is 0.5 * 0.14 = 0.07

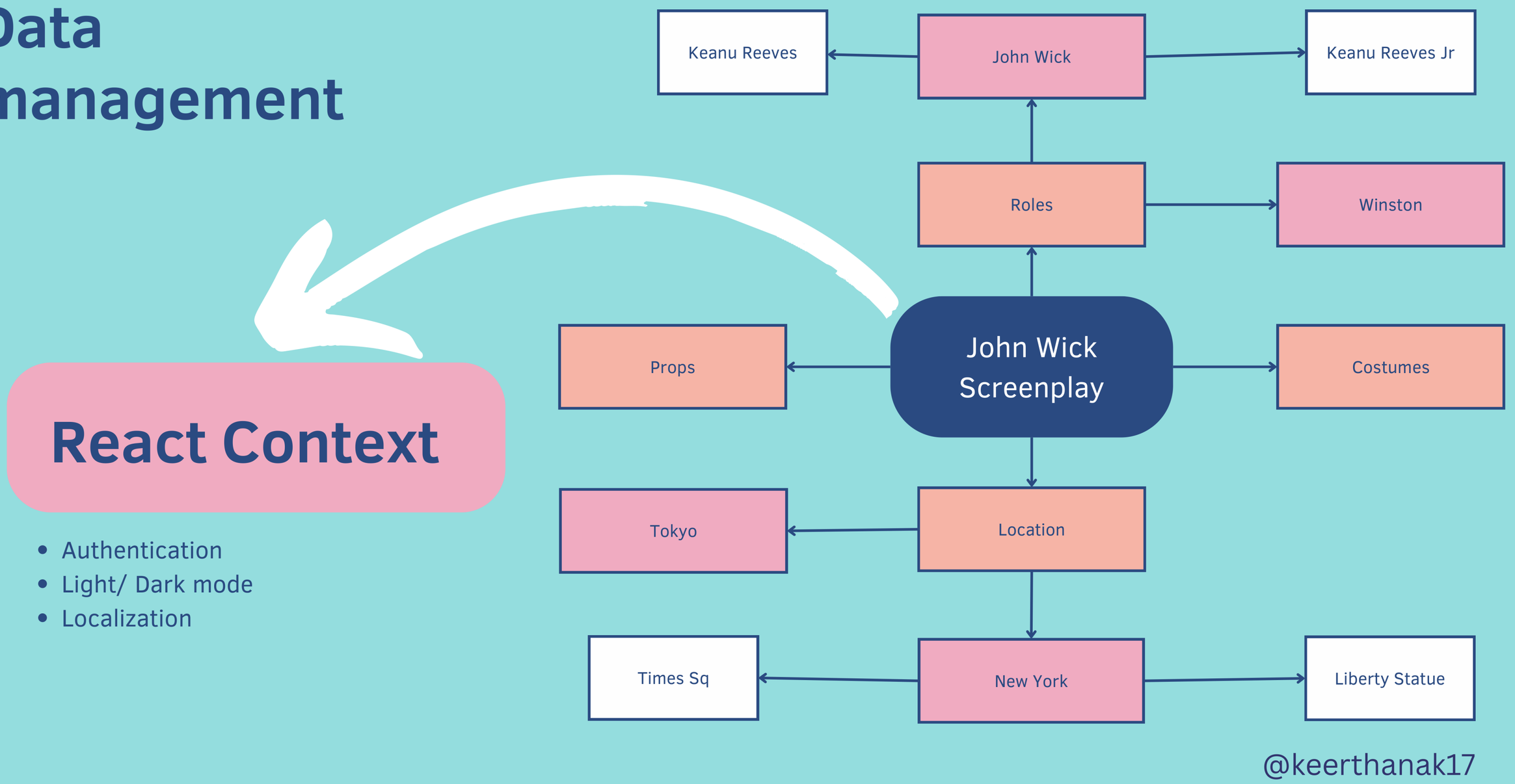
@keerthanak17

CLS Example with multiple unstable elements



- The "Cat" item is the only one from the first screen which is stable
- The items “Crocodile”, “Elephant”, “Lion” and “Tiger” are new in the DOM, so their start position doesn't change, its not unstable
- The items “Dog”, “Horse” and “Elephant” all shift their start positions, making them unstable
- The layout score considers the fraction of the viewport impacted (60%, red dashes) and the longest fraction of the viewport distance that an element moved (30%, blue arrow)
- Layout shift score is 0.6 * 0.3 = 0.18

Data management



React Context

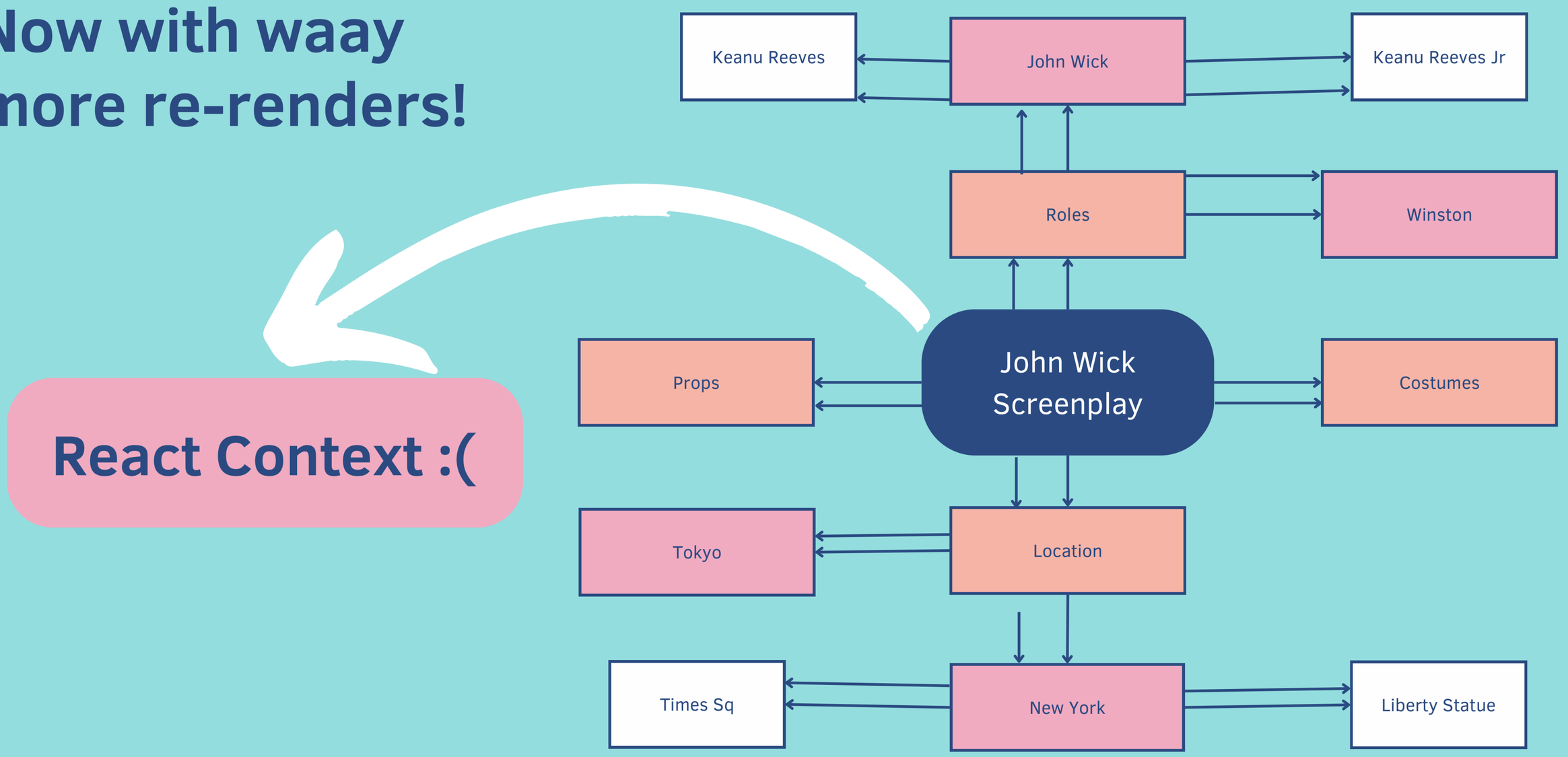
- Authentication
- Light/ Dark mode
- Localization

@keerthanak17

But we want script versions!

ALSO SCRIPT EDITING :)

Now with waay more re-renders!



@keerthanak17

How a site functions in development is often quite different from how users experience it

Third-party content or test images are often already in the developer's browser cache, and API calls that run locally are often so fast that the delay isn't noticeable.

Common causes of CLS

1 — 2 — 3 — 4

STEP

Images without dimensions

Always include width and height size attributes on your images and video elements or aspect-ratio for responsive content

STEP

Ads, embeds, and iframes

Statically reserve space for late-loading content. Use *min-height* to reserve space or *aspect-ratio* for responsive content

STEP

Changing top/left animations

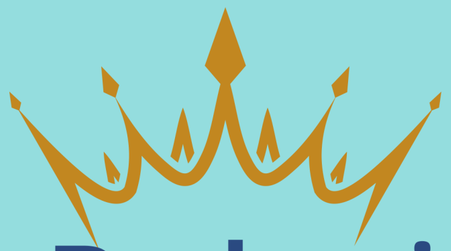
Animations with transform can translate, scale, & more without triggering a re-layout and so completely avoiding layout shifts

STEP

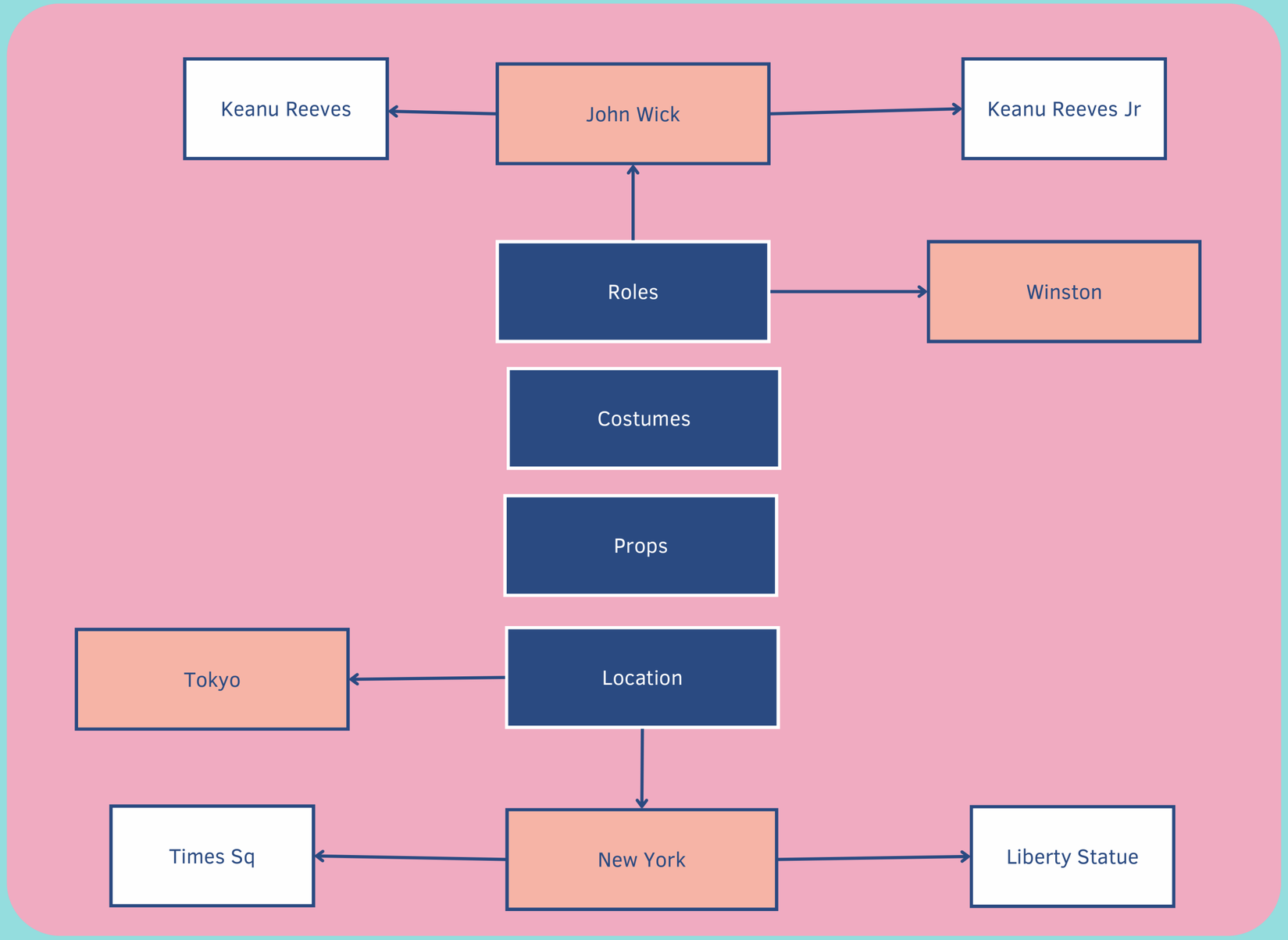
Loading a web font improperly

Manage the fallback font properly to avoid re-layout as the web font(s) are loaded since both of these can cause layout shifts

@keerthanak17



Redux is here!



@keerthanak17

More scenarios

INSTANCES IN YOUR APPLICATION WHICH COULD IMPACT YOUR CLS SCORE

Displaying a (large) collection of data

Manage your lists or tables with skeletons and virtualization to minimize the layout shifts on data load, opposed to multiple circular loaders

---

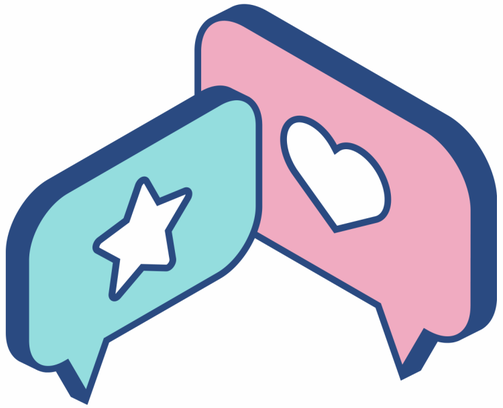
Managing error conditions

Make sure that your error boundaries are designed to minimize any layout shifts for the user

---

Handling non discrete input events

Continuous interactions such as scrolls, drags, or pinch and zoom gestures are not considered "recent input". See the Layout Instability Spec for more details.



Interaction to Next Paint (INP)

ASSESSES THE RESPONSIVENESS THROUGHOUT THE PAGE LIFECYCLE

- Calculated when the user leaves the page, resulting in a single value that is representative of the page's overall responsiveness throughout the entire page's lifecycle.

- Considers discrete input events, like clicking a screen, tapping a touch screen or pressing key in a keyboard. Hovering and scrolling, even if it’s done using keystrokes are not considered in the final score calculation

- It may not exist if the user doesn’t click or tap on any content item, they simply scrolled or hovered over elements, if the page is accessed by a bot such as a search crawler or headless browser

@keerthanak17

FID vs INP

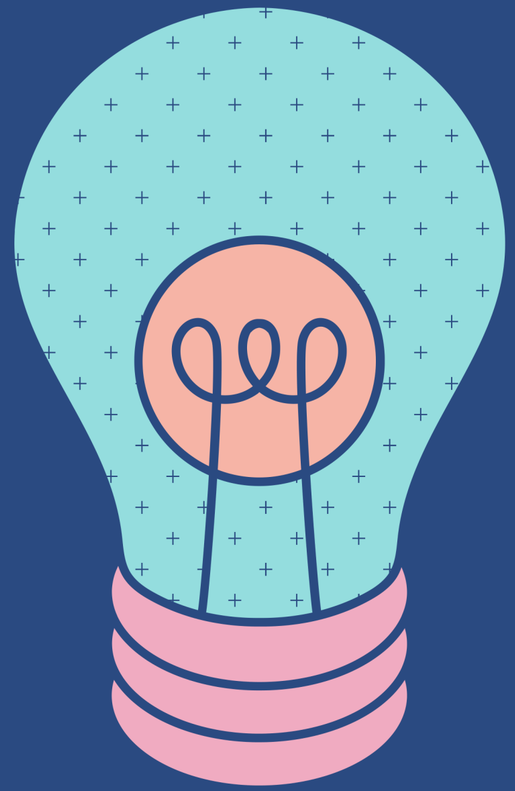
HOW DOES THE DIFFERENCE MATTER?

What is considered?
Where INP considers all page interactions, First Input Delay (FID) only accounts for the first interaction
What FID looks for
FID is also a load responsiveness metric, believing that if the first interaction made with a page in the loading phase has little to no perceptible input delay, the page has made a good first impression.
INP's point of focus
INP is more than about first impressions. By sampling all interactions, responsiveness can be assessed comprehensively, making INP a more reliable indicator of overall responsiveness than FID

@keerthanak17

“FID didn’t vary across frameworks. For React, LCP remains mostly the same, while CLS actually degrades, compared to Vue"

WHAT HAPPENS WHEN INP IS CONSIDERED?



@keerthanak17

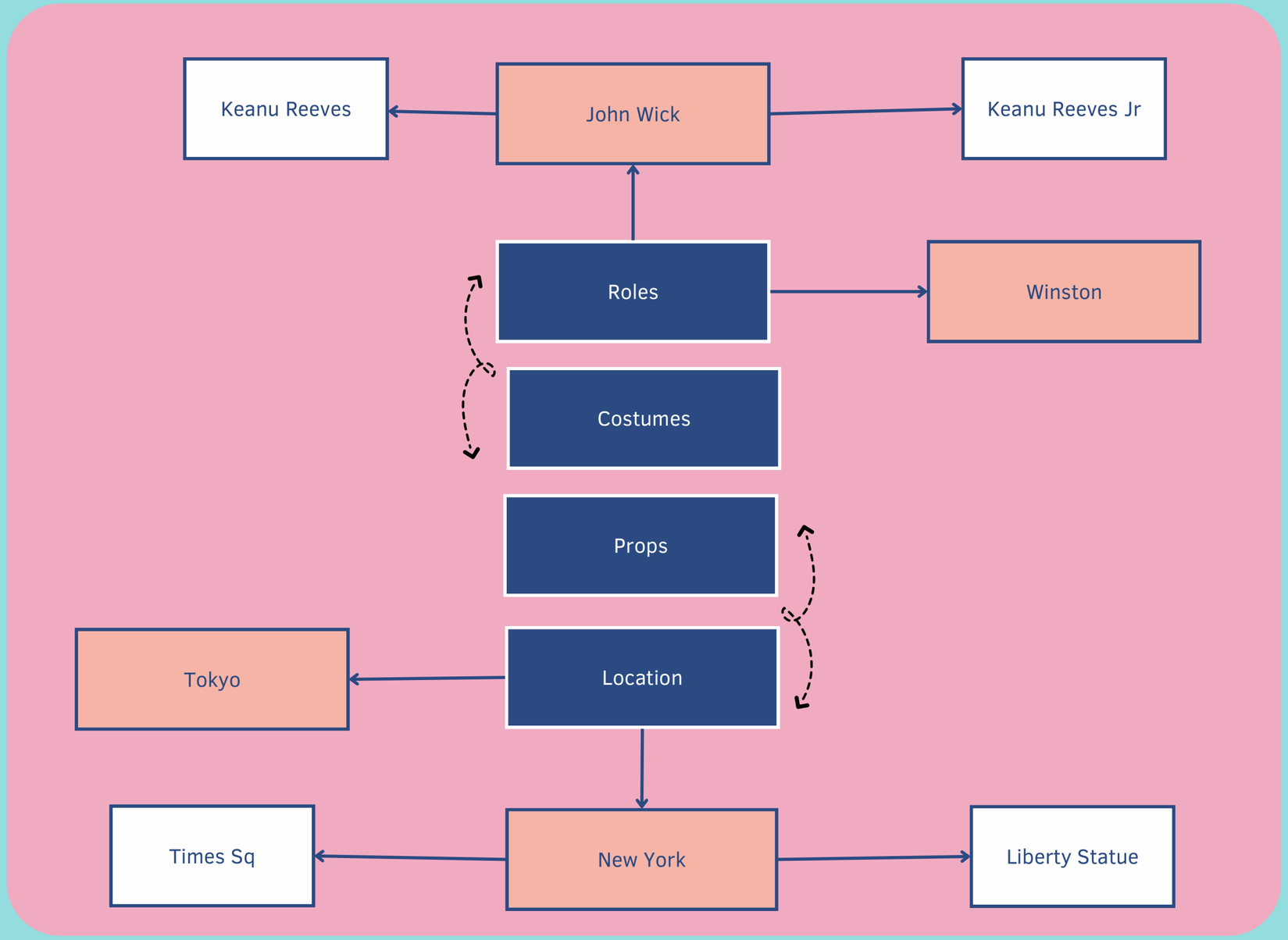
Re-rendering, AKA, the final boss

VUE CONTROLS BOTH THE COMPILER AND THE RUNTIME ALLOWING THE ADVANTAGE COMPILE-TIME OPTIMIZATIONS

In React, even if a part of the tree never changes, new vnodes are always created for them on each re-render, resulting in unnecessary memory pressure. The somewhat brute-force reconciliation process sacrifices efficiency in return for declarativeness and correctness.

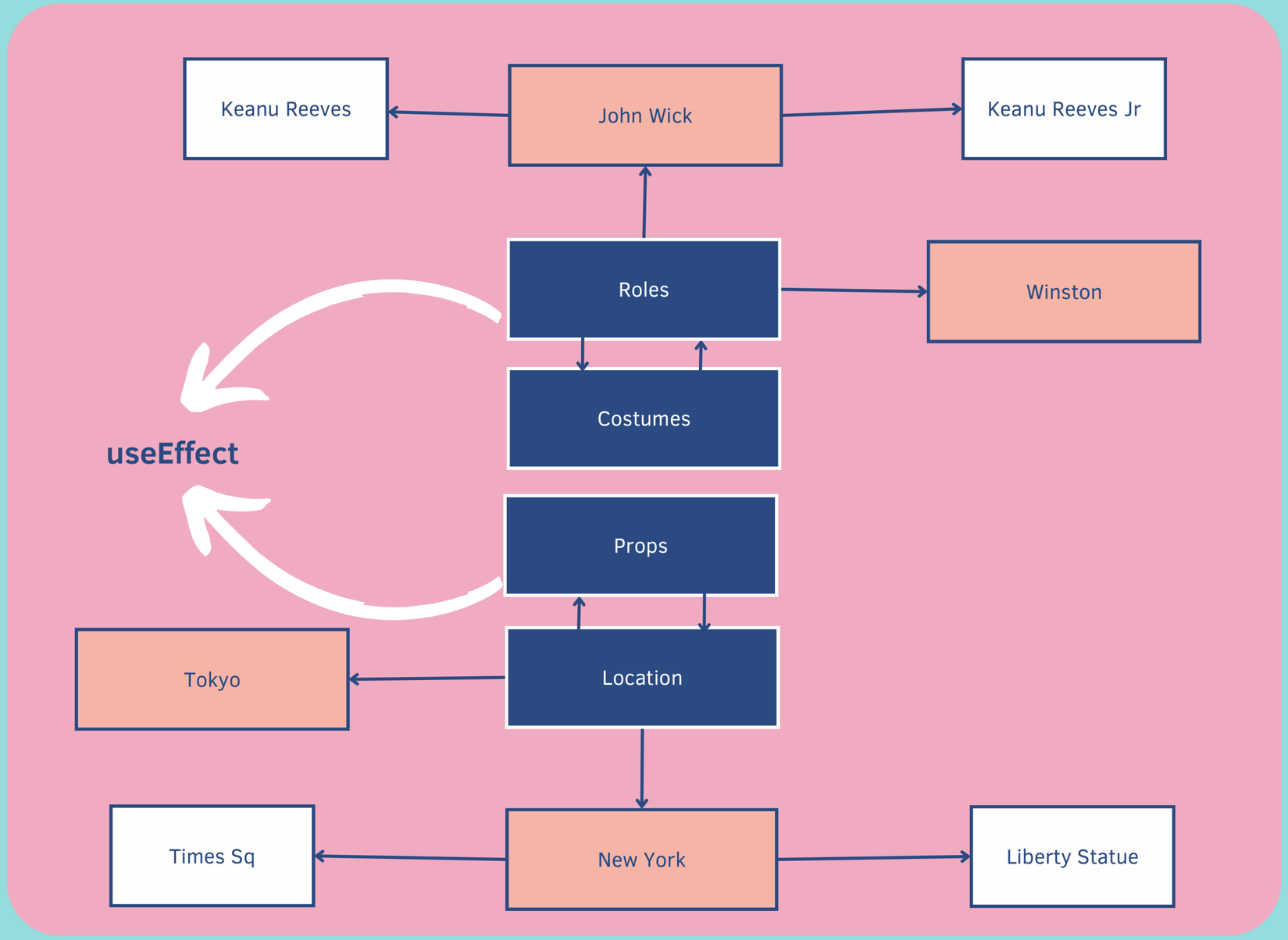
@keerthanak17

Oh no,
dependencies!



@keerthanak17

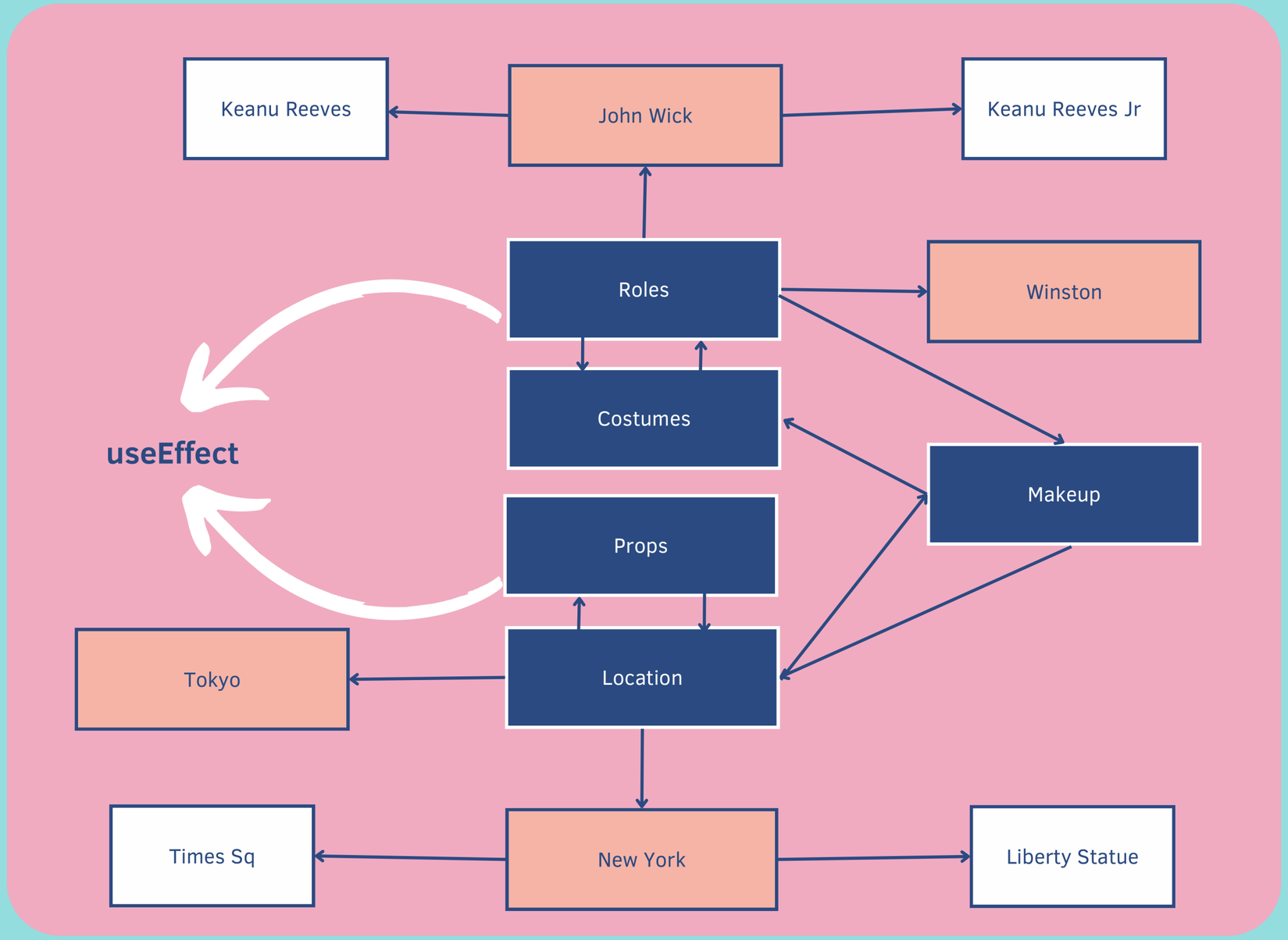
Handling inter-dependance with a useEffect()

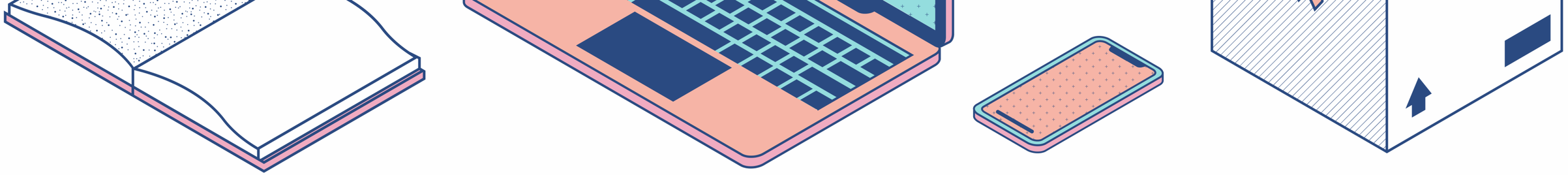


```
useEffect(()=>{
refreshCostumes()
},[roleId])
```

```
useEffect(()=>{
refreshProps()
},[locationId])
```

Complexity


Keanu Reeves
John Wick
Keanu Reeves Jr
Roles
Winston
Costumes
useEffect
Makeup
Props
Tokyo
Location
Times Sq
New York
Liberty Statue

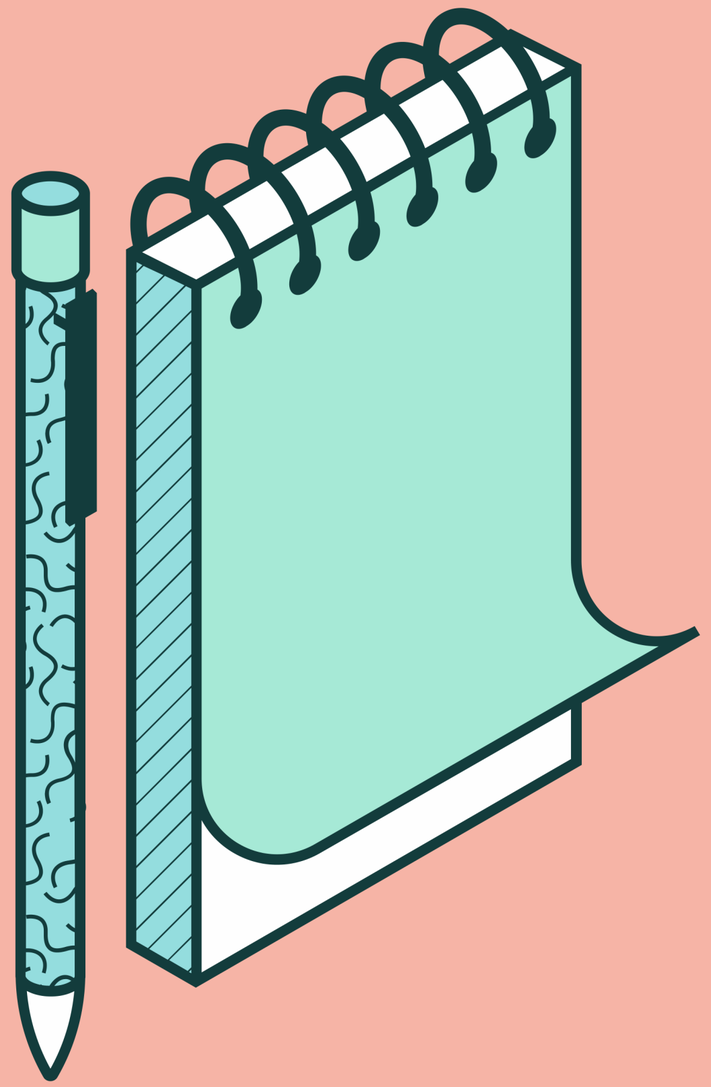


Layouts & style calculations

- Layout has a direct effect on interaction latency
- Layout is normally scoped to the whole document.
- The number of DOM elements will affect performance; you should avoid triggering layout wherever possible

Thrashing

- Forcing style calculations synchronously in succession
- Running the style calculations and layout synchronously and earlier than the browser would like are potential bottlenecks
- Use the profiler to see the layout calculation breakdown



useLayoutEffect()

A VERSION OF THE USEEFFECT()
THAT FIRES BEFORE THE BROWSER
REPAINTS THE SCREEN

EX: TOOLTIPS

@keerthanak17

React.memo()

REDUCING RE-RENDERS
OF FUNCTIONAL
COMPONENTS

Skip component rerendering

With memo(), components do not re-render with their parents as long as the props, state or context are the same

---

Customization with optional arePropsEqual()

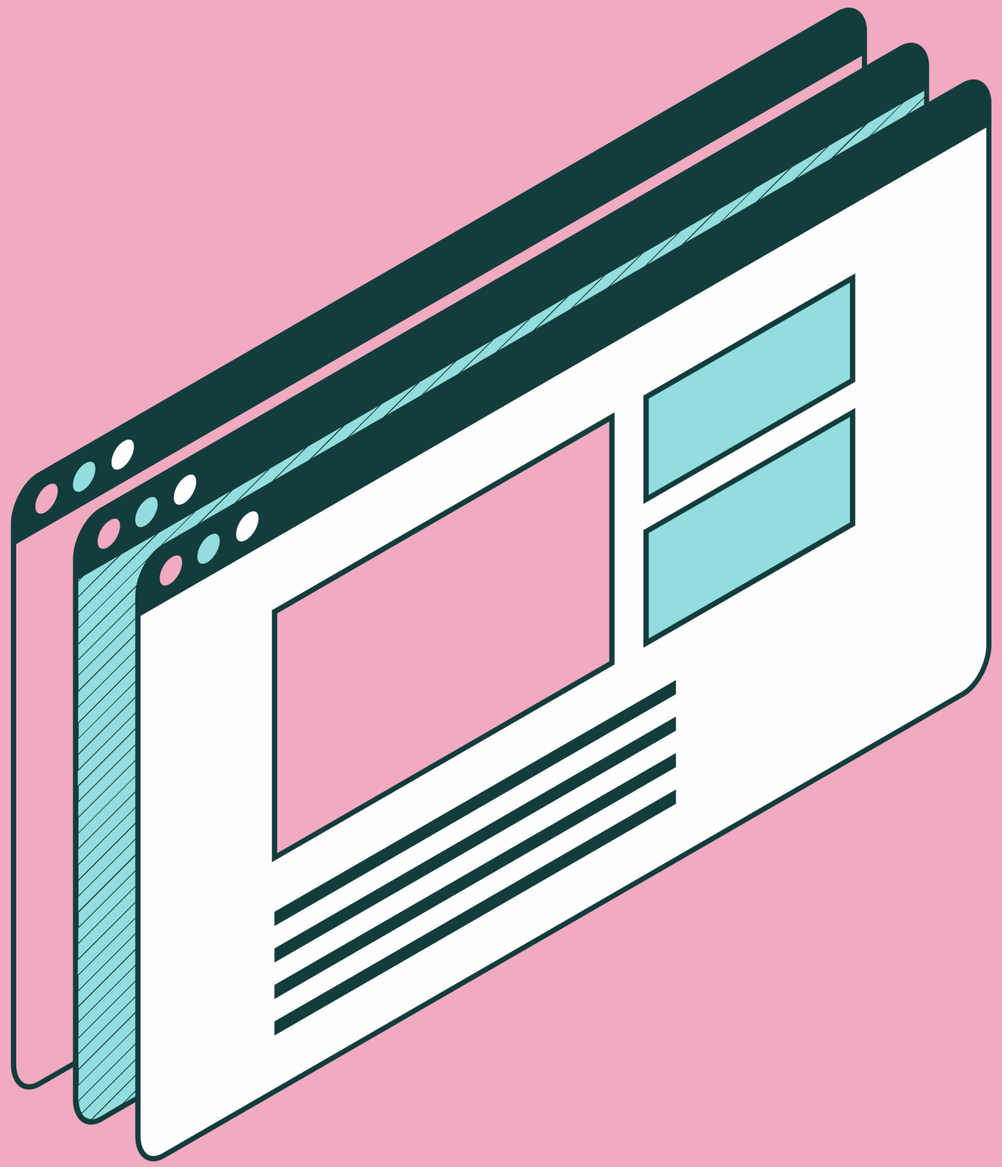
It lets you define a custom function for checking the difference between props which should return a boolean value

---

Prop is an array, object or function

Default prop comparison uses shallow equality. Simplify or memoize props in the parent component

@keerthanak17



We want different views of the data

LIST, THUMBNAILS, DETAILS

@keerthanak17

```
const Costumes = React.memo(
    () => {
        // show costumes for a role
    },
    function (prevProps, nextProps) {
        prevProps.roleId === nextProps.roleId &&
        prevProps.locnId === nextProps.locnId
    }
)
```

useCallback()

REDUCING RE-RENDERS
OF FUNCTIONS

Cache a function definition between re-renders

Ensure that it’s the same function between re-renders when the dependencies do not change

---

Parameters of a useCallback()

It takes any function and a list of dependencies and returns a function which is cached till any of its dependencies change

---

No conditional rendering

Like any React hook, useCallback() can only be called at the top level of a component and not with any loops or conditions

@keerthanak17

```
const ActorProfile = ( actorId, actorCountry) => {
    const sizeConverter = useCallback(
        (costumeList) => {
            // Convert sizes
        },
        [ actorId, actorCountry]
    )

}
```

useMemo()

REDUCING RE-RENDERS
OF VALUES

Minimize redundant calculations

Used to cache the results of an expensive calculation between re-renders of a component

---

Parameters of a useMemo()

It has a pure function to calculate the value you want to cache and a dependency array of all reactive values which were used

---

useMemo for items in a list

To avoid layout thrashing, useMemo should be used inside a component extracted for each item in a list

@keerthanak17

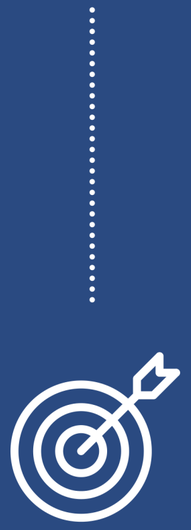
```
const RoleDetails = ({ costumeList, makeupList, roleData }) => {
    const roleBudget = useMemo(() => {
        // Sum of all costumes and makeup costs
    }, [costumeList, makeupList])

    return <>{roleBudget}</>
}

const RoleList = ({ costumeList, makeupList, roles }) => {
    return (
        <>
            {roles.map((role) => (
                <RoleDetails
                    costumeList={costumeList}
                    makeupList={makeupList}
                    roleData={role}
                />
            ))}
        </>
    )
}
```

Dependencies

KEEP YOUR EYES ON IT

@keerthanak17

This is just the starting point



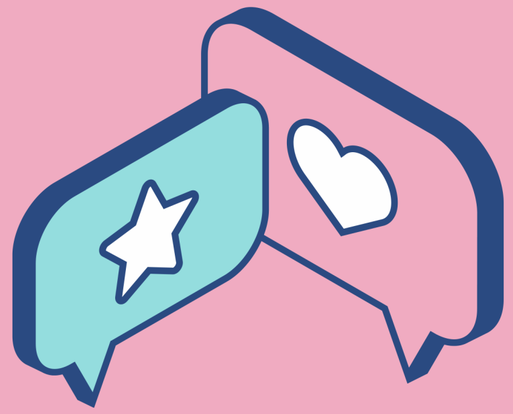
Look at the challenges unique to your project and see how these principles apply

@keerthanak17

Do you have any questions?

Speak now or forever hold your peace (or, you can google it?)

@keerthanak17



@keerthanak17

Appendix I

Links to external resources for further reading

@keerthanak17

- Core Web Vitals - https://web.dev/articles/vitals
- An In-Depth Guide To Measuring Core Web Vitals - https://www.smashingmagazine.com/2021/04/complete-guide-measure-core-web-vitals/
- How To Use Google CrUX To Analyze And Compare The Performance Of JS Frameworks - https://www.smashingmagazine.com/2022/05/google-crux-analysis-comparison-performance-javascript-frameworks/
- **Real World Problems - Internet beyond the 1st World Bubble | JSConf.Asia 2019 - https://www.youtube.com/watch?v=skml8SjOtKg**
- Replacing FID - https://developers.google.com/search/blog/2023/05/introducing-inp
- About INP - https://web.dev/blog/inp-cwv