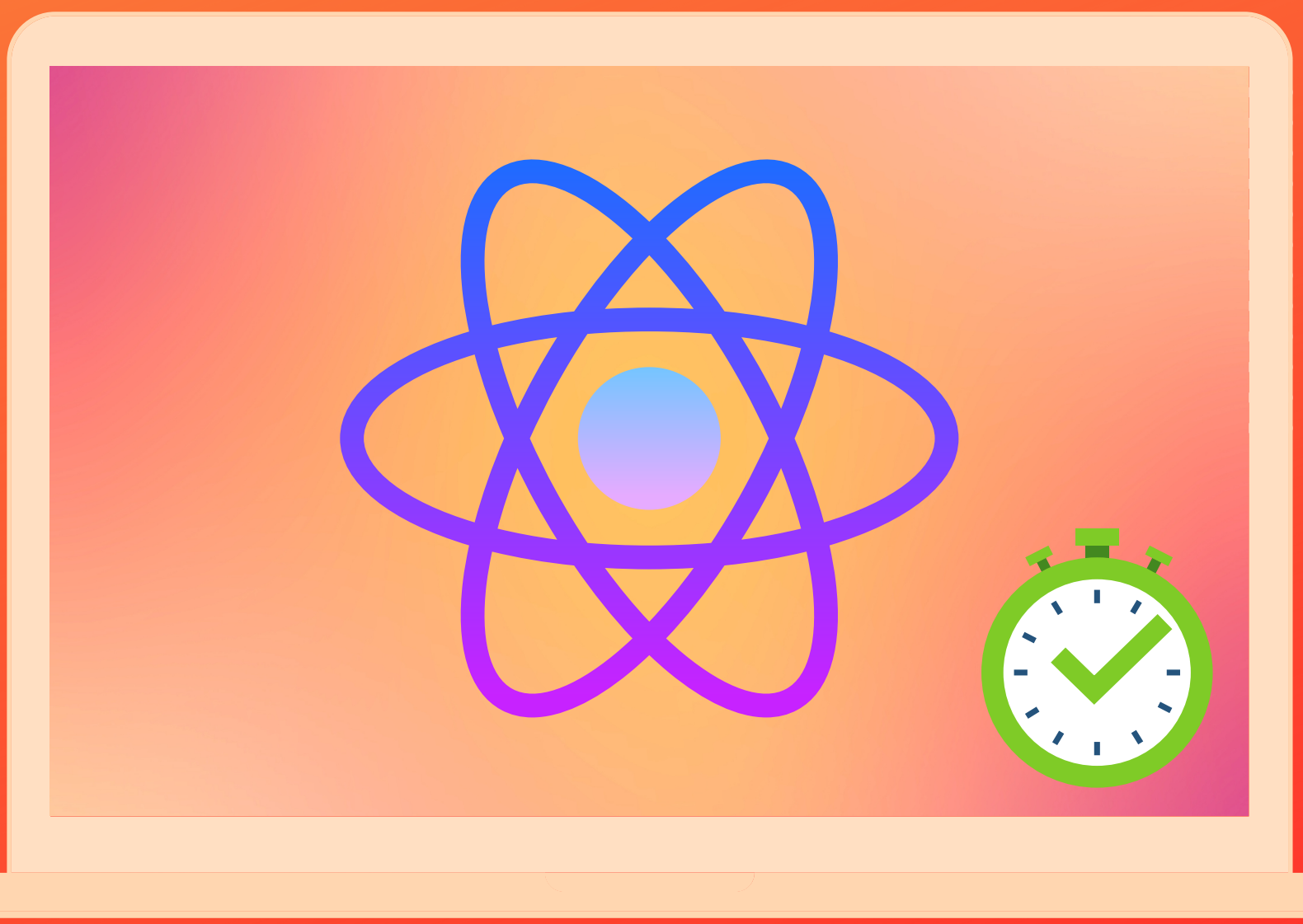
Effective ways to optimize React App's Performance



@srikargvs17

Optimizing Performance of React web applications is always challenging and resolving issues has become a pain area for developers in most of the projects they are working for

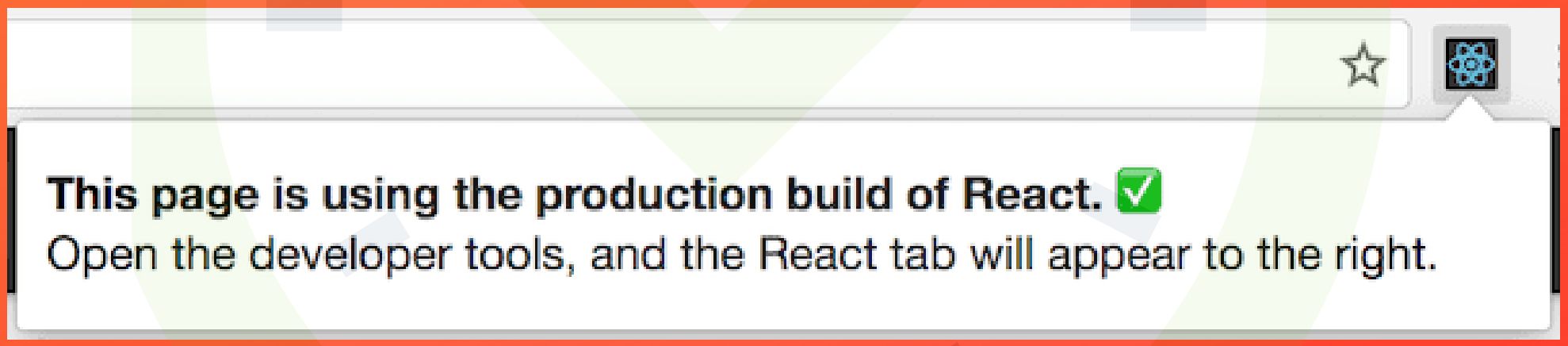
@srikargvs17

Though React uses several clever techniques internally to optimize performance, below are few useful techniques in brief that every React Developer should be aware of to achieve better performance...

@srikargvs17

Always use Production build

Always deploy production build to customer facing servers. This will help in removing features specifc to development mode like "warnings". ***React-Dev-Tools*** *(browser extension) can help to check if you are on production build..*



Tip: Use module bundlers like Webpack to build in production mode and add compression plugins using gzip/brotli algorithms

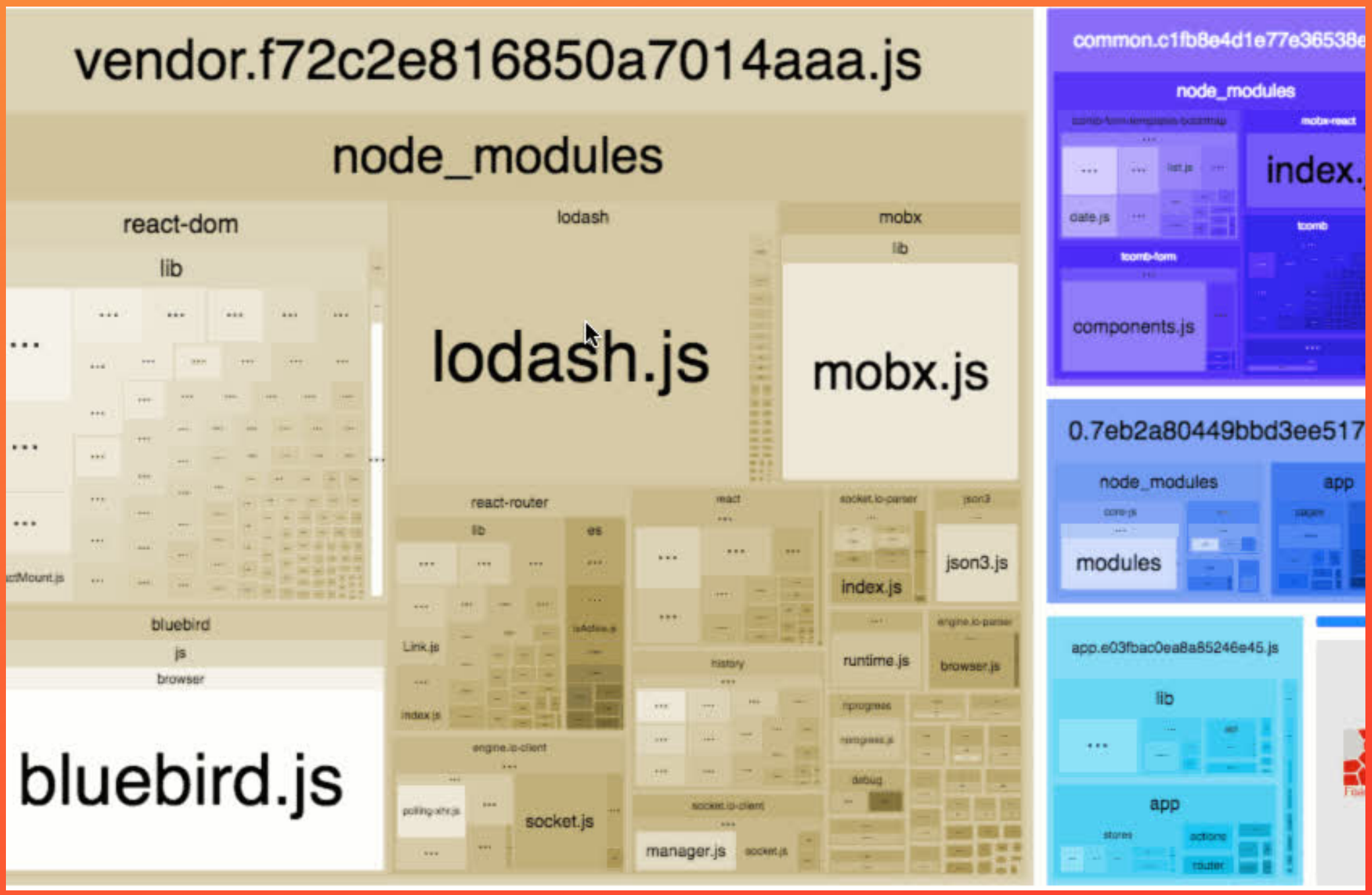
Analyzing Production Bundle

*Install **BundleAnalyzer** Plugin for Webpack and analyze sizes of various files contributing to your final production package and work on them to reduce its size*

npm install --save-dev webpack-bundle-analyzer

```
const BundleAnalyzerPlugin = require('webpack-bundle-analyzer').BundleAnalyzerPlugin;

module.exports = {
  plugins: [
    new BundleAnalyzerPlugin()
  ]
}
```

@srikargvs17


vendor.f72c2e816850a7014aaa.js
node_modules
react-dom
lib
lodash
lodash.js
mobx
lib
mobx.js
bluebird
js
browser
bluebird.js
react-router
lib
es
Link.js
index.js
socket.io-client
socket.js
manager.js
react
history
socket.io-parser
index.js
runtime.js
json3
json3.js
engine.io-parser
browser.js
debug
common.c1fb8e4d1e77e36538e
node_modules
components.js
index.
0.7eb2a80449bbd3ee517
node_modules
modules
app
app.e03fbac0ea8a85246e45.js
lib
app
stores
actions
router

Code Splitting

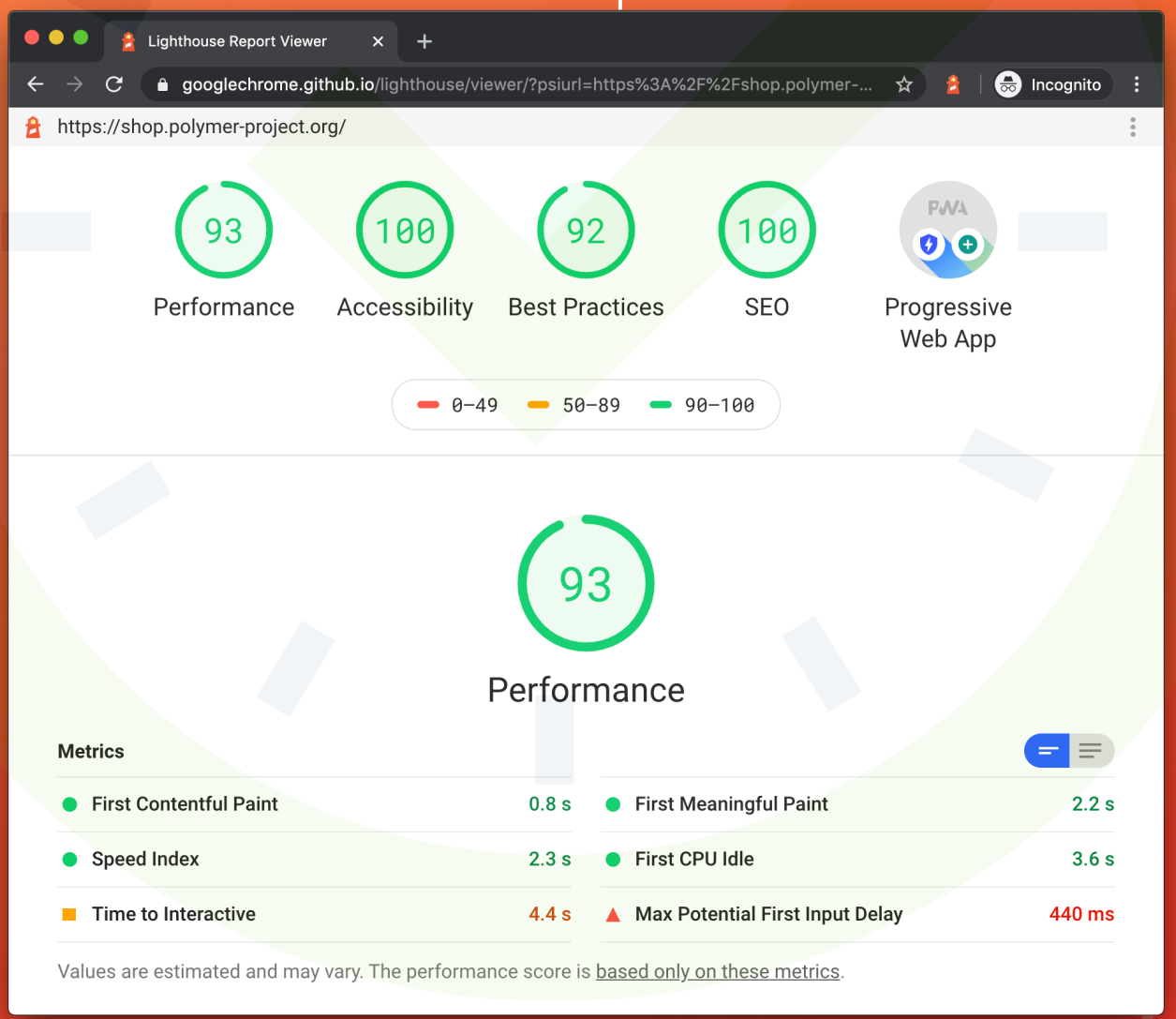
Using React.lazy & Suspense , load only components on a webpage that are needed

```jsx
import React, { Suspense } from 'react';

const OtherComponent = React.lazy(() => import('./OtherComponent'));

function MyComponent() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <OtherComponent />
      </Suspense>
    </div>
  );
}
```

Check Lighthouse Score and implement suggestions

You can run Lighthouse in Chrome DevTools

- Chrome Dev Toos > Switch to Light hosue > Generate Report



@srikargvs17

Stop re-rendering components using React.memo

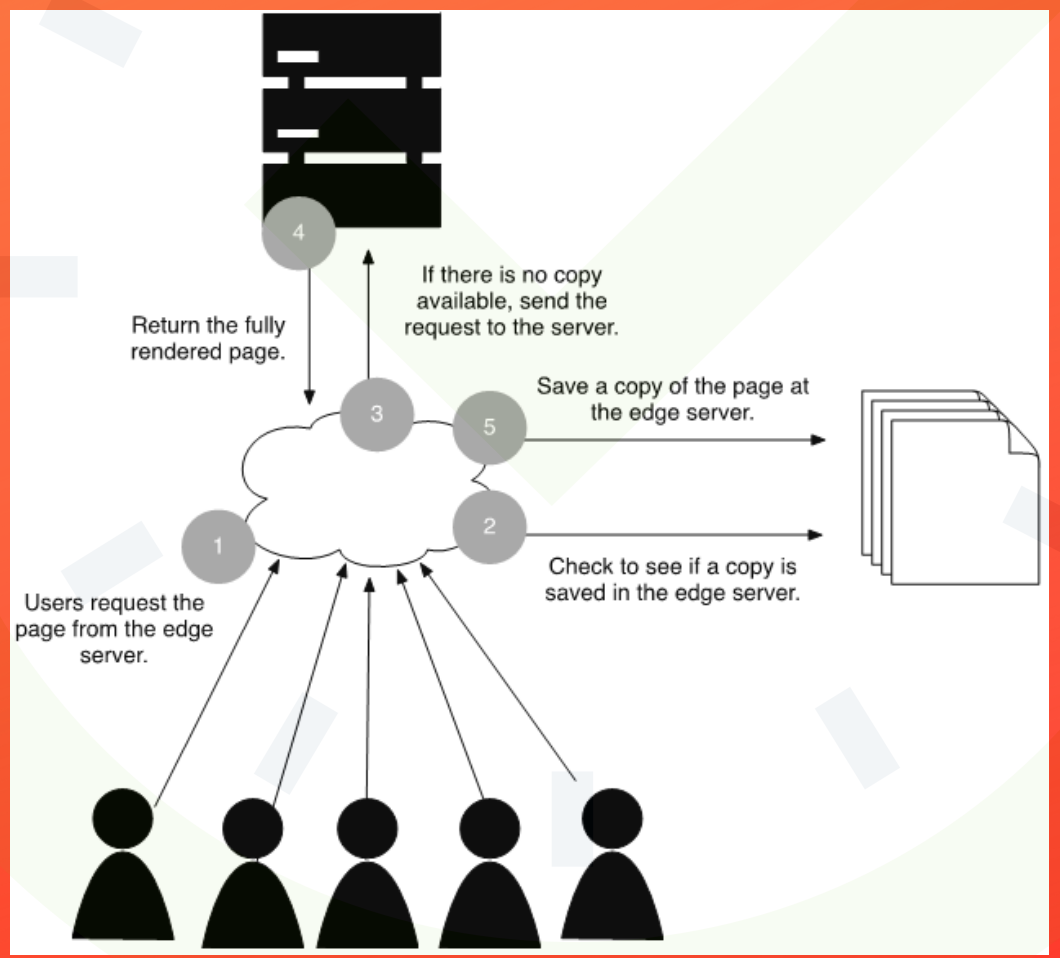
- React.memo only checks for prop changes. If your function component wrapped in React.memo has a useState, useReducer or useContext Hook in its implementation, it will still rerender when state or context change.Memo

By default it will only shallowly compare complex objects in the props object

```
function MyComponent(props) {
  /* render using props */
}
function areEqual(prevProps, nextProps) {
  /*

  return true if passing nextProps to render would return
  the same result as passing prevProps to render,
  otherwise return false

  */}
export default React.memo(MyComponent, areEqual);
```

Using CDN Caching techniques

- CDN stands for Content Delivery Network
- If user requests for a webpage, request to hosting server will be made if and only if, page is not cached or available on CDN/Edge servers



…. and the list/techniques continues to grow. Trying above will definitely otptimize the performance and improve the scores.

Bonus

- React Window to optimize long lists.
- Replacing inline arrow functions in render method.
- In-memory caching.
- Debouncing JavaScript events.

Info Credits:

- ***https://reactjs.org/docs***
- ***https://www.npmjs.com/package/webpack-bundle-analyzer***
- ***https://developers.google.com/web/tools/lighthouse/***
- ***https://freecontent.manning.com/caching-in-react/***

@srikargvs17

Thanks for reading!!

CHEERS!

@srikargvs17

@srikargvs17