



# Debajit Mallick

## Software Engineer @P360



# who am I?

- Organizer @GDG Siliguri
- LinkedIn 2024 Top Voice for Web Development.
- Mentor and Judge of Hack4Bengal 3.0 and 2.0 Hackathon.
- Mentor of team OrganiCod3rs, the Winner of Smart India Hackathon 2022, Software Edition.
- Mentor of GirlScript Summer of Code 2023.
- Member of Team Delenitors, Smart India Hackathon 2020 Winner, Software Edition.
- Top Contributor of GirlScript Winter of Code 2021.
- Top Contributor of JGEC Winter of Code 2020





**CHALIYE  
SHURU  
KARTE HAI**



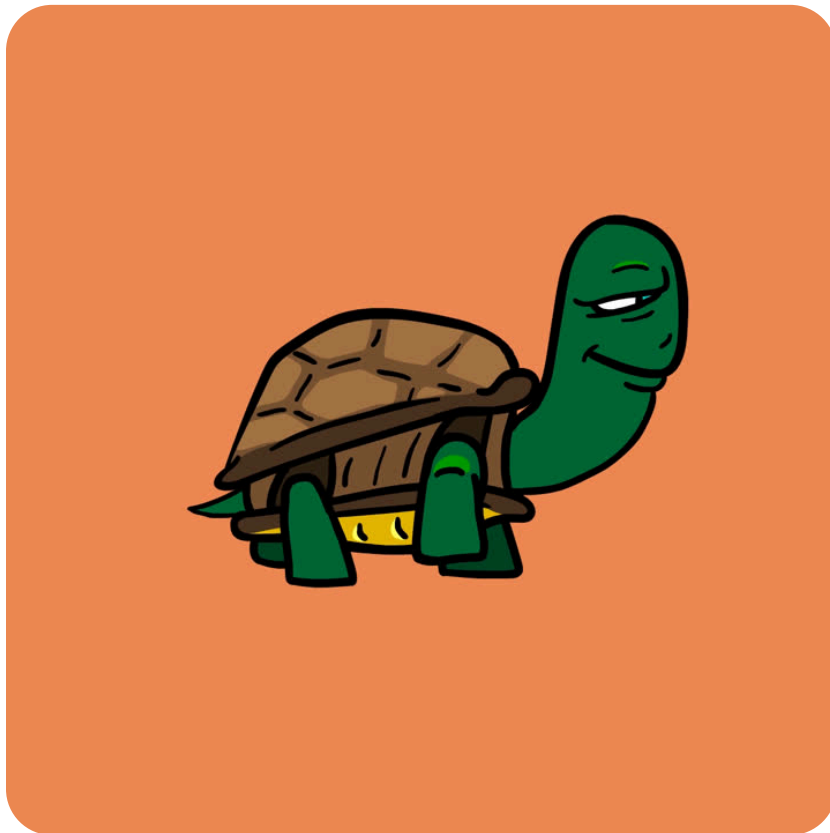


# Topic

**Performance-Driven Web Development:  
Mastering Speed and Optimization**



# Webapps



**slow**



**fast**







# How to become a performance focused developer?

- Image Optimization
- Code Splitting
- Memoization
- Debouncing and Throttling
- Caching
- Adaptive Loading





yeh sab kya tha



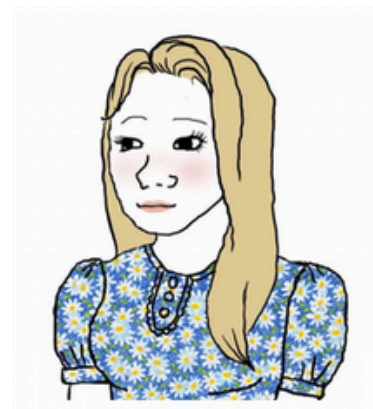


# 1. Image Optimization

Everyones biggest fear



\*asks her out\*



no

## Normal People

Web Performance's biggest fear



images

## Developers



# Things to keep in mind while handling images

## 1. Implement Lazy Loading for images

```
<div>  
    
</div>
```

## 2. Using CDNs

CDNs distribute the images across multiple servers worldwide, reducing latency and improving loading speeds for users from different regions.



### 3. Using correct format for images

Consider using the modern WebP or AVIF image format, which provides better compression and quality compared to JPEG and PNG formats. However, ensure to provide fallbacks for browsers that don't support WebP or AVIF.

### 4. Using Proper compression

Always compress your images to reduce their file size without losing quality. You can use various tools and libraries like [imagemin](#) or [TinyPNG](#) to compress images before including them



**\*To learn more about image optimization check out this blog:**







# Code Splitting

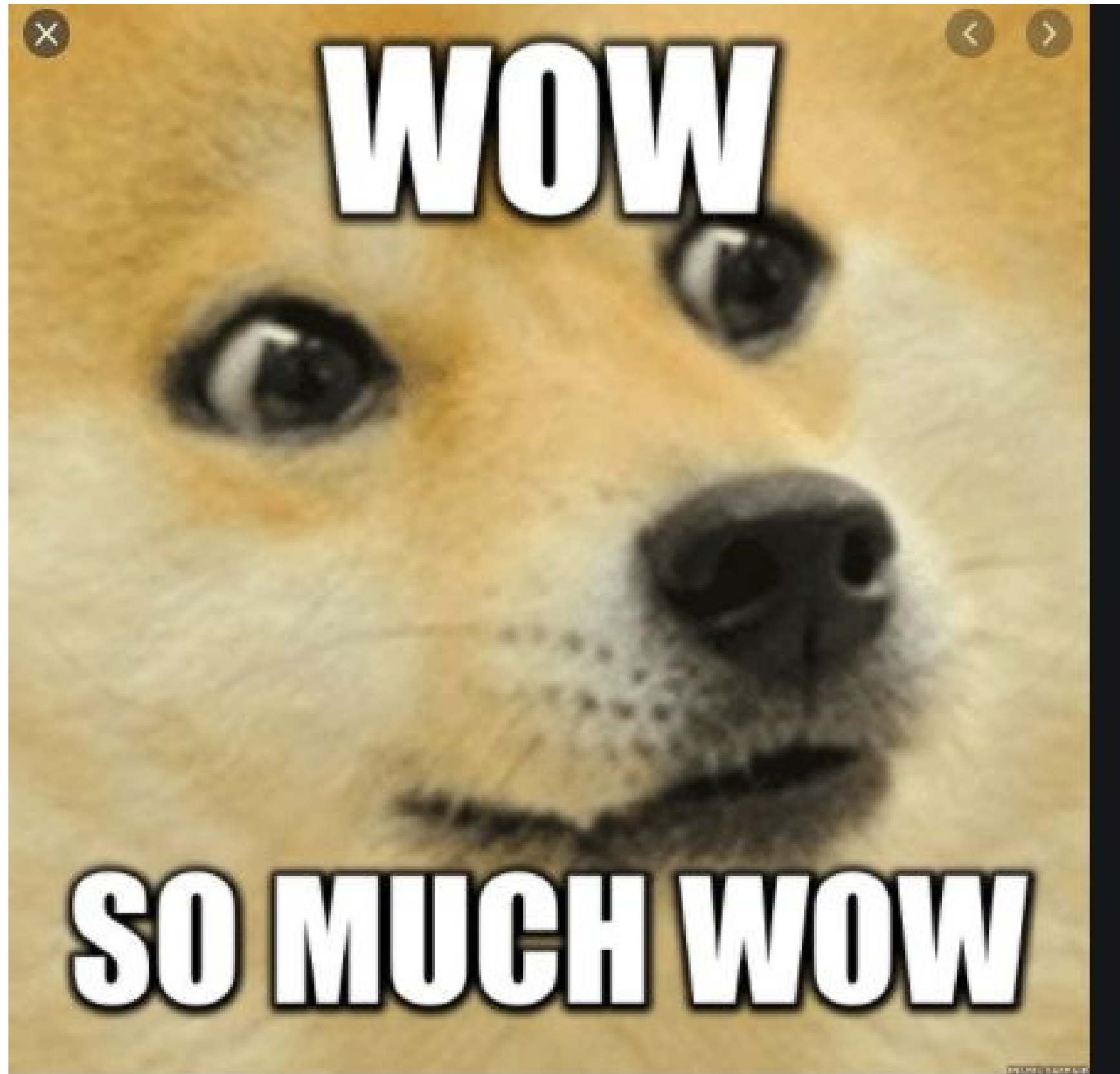
Code Splitting is one of the most popular techniques for optimizing web performance. In Code Splitting, we split our application into smaller chunks and load them when needed. `<Suspense/>` and `lazy()` will help us with this.

```
import React, { Suspense } from 'react';
const OtherComponent = React.lazy(() => import('./OtherComponent'));

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







# Memoization

Memoization is one of the amazing features of React and other modern frameworks. State and function re-rendering is an expensive computational task. To reduce unnecessary re-renders we can use the `useMemo` and `React.memo` functions.

```
const MyComponent = React.memo(TestComponent);
```



# Debouncing and Throttling

Specially for scroll and resize events we do a lot of re-rendering in React. Also, for inputs onChange events state change is a computation-intensive task. Implement debouncing and throttling for event handlers to prevent excessive rendering.

```
const App = () => {
  const [name, setName] = useState('');
  const [debouncedValue, setDebouncedValue] = useState('');

  useEffect(() => {
    const timeoutId = setTimeout(() => {
      setDebouncedValue(value);
    }, 2000);

    return () => clearTimeout(timeoutId);
  }, [value]);

  return (
    <input onChange=((e) => {
      setName(e.target.value);
    })/>
    <p>Debounced Value: {debouncedValue}</p>
  );
}
```



# Caching

Caching is one of the most underrated ways to improve performance. Implement client-side caching for data that doesn't change frequently to reduce the need for unnecessary API calls.

```
function cacheData(key, data) {  
  localStorage.setItem(key, JSON.stringify(data));  
}
```



**\*To learn more about performance optimization check out this blog:**



# Adaptive Loading





# Adaptive Loading

## Render different components based on

- Network status - (5g/4g/3g)
- Save Data Preferences
- Media Capabilities
- CPU Cores / Hardware Concurrency



**\*To learn more about adaptive loading check out this blog:**



**Now, I am a Performance Guru**

**PPT Link**



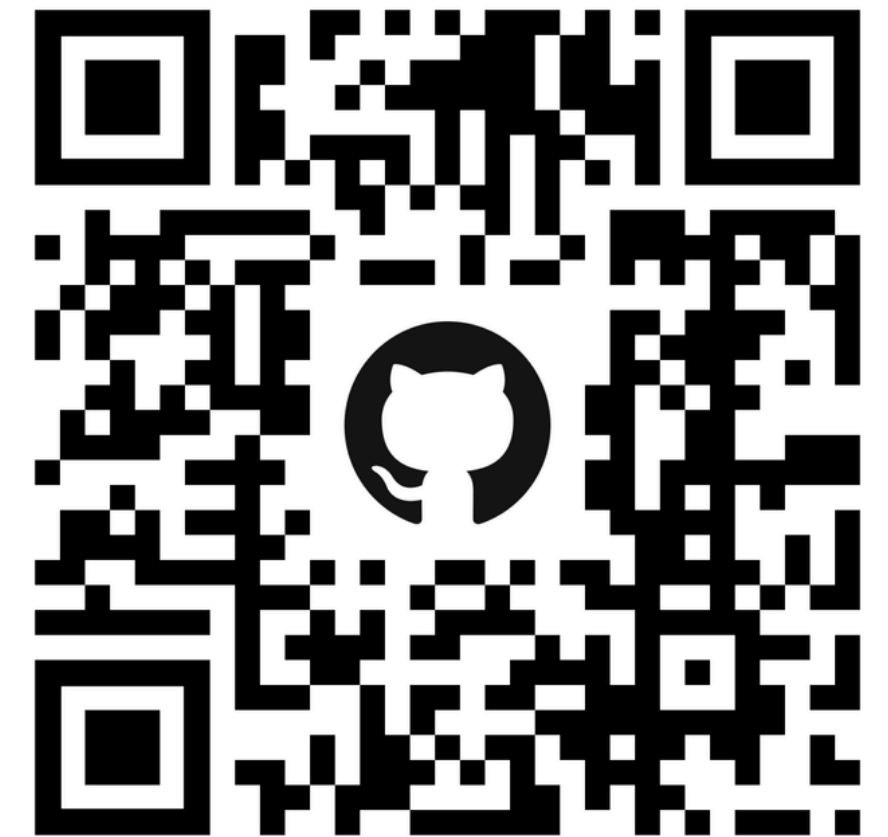
# Thank You



[x.com/MallickDebajit](https://x.com/MallickDebajit)



[linkedin.com/in/debajit-mallick/](https://linkedin.com/in/debajit-mallick/)



[github.com/debajit13](https://github.com/debajit13)

