1.

```
export class ColorFromImgDirective implements OnChanges {
  @Input()
  src: string;

  @Output()
  newColor = new EventEmitter();

  worker: Worker;
  image: Partial<CanvasImage>;
  private imgSrc = "";
  constructor() {
    this.worker = new Worker(new URL("./quantize.worker", import.meta.url));
    this.worker.onmessage = ({ data }) => {
      this.image.removeCanvas();
      this.newColor.emit(data);
    };
  }
  ngOnChanges({ src }: SimpleChanges) {
    if (src.currentValue !== "assets/imgs/default.svg") {
      if (src.firstChange) {
        this.imgSrc = src.currentValue;
        this.getPaletteFromUrl(this.imgSrc);
      } else {
        const incomingVal = new URL(src.currentValue);
        const currentVal = new URL(src.previousValue);
        if (currentVal.pathname !== incomingVal.pathname) {
          this.imgSrc = src.currentValue;
          this.getPaletteFromUrl(this.imgSrc);
        }
      }
    }
  }

  getPalette(
    sourceImage: HTMLImageElement,
    colorCount?: number,
    quality?: number
  ) {
    if (
      typeof colorCount === "undefined" ||
      colorCount < 2 ||
      colorCount > 256
    ) {
      colorCount = 10;
    }
    if (typeof quality === "undefined" || quality < 1) {
      quality = 10;
    }

    // Create custom CanvasImage object
    this.image = canvasImage(sourceImage);
    const imageData = this.image.getImageData();
    const pixels = imageData.data;
    const pixelCount = this.image.getPixelCount();
    this.worker.postMessage({ pixels, pixelCount, colorCount, quality });
  }

  getPaletteFromUrl(imageUrl: string, quality = 10) {
    new Promise((resolve, reject) => {
      const sourceImage = new Image();
      sourceImage.crossOrigin = "Anonymous";
      sourceImage.addEventListener("load", () => {
        const palette = this.getPalette(sourceImage, 5, quality);
        resolve({ palette, imageUrl });
      });
      sourceImage.src = imageUrl;
      sourceImage.addEventListener("error", reject.bind(this));
    });
  }
}
```