



```
OInput()
    src: string:
 3
 4
    @Output()
5
6
    newColor = new EventEmitter();
 7
8
    worker: Worker:
9
    image: Partial<CanvasImage>:
    private imgSrc = "":
10
    constructor()
11
12
      this.worker = new Worker(new URL("./quantize.worker", import.meta.url));
13
      this.worker.onmessage = (( data |) \Rightarrow (
        this.image.removeCanvas();
14
15
        this.newColor.emit(data);
16
17
    ngOnChanges ( src ): SimpleChanges) {
18
      if (src.currentValue ≠ "assets/imgs/default.svg") {
19
        if (src.firstChange) {
20
21
          this.imgSrc = src.currentValue;
22
          this.getPaletteFromUrl(this.imgSrc):
23
        else
24
          const incomingVal = new URL(src.currentValue);
25
          const currentVal = new URL(src.previousValue);
26
          if (currentVal.pathname == incomingVal.pathname) {
27
            this.imgSrc = src.currentValue;
28
            this.getPaletteFromUrl(this.imgSrc);
29
39
31
32
33
34
    getPalette(
35
      sourceImage: HTMLImageElement,
36
      colorCount?: number.
37
      quality?: number
38
39
      if (
40
        typeof colorCount == "undefined" ||
41
        colorCount < 2 ||
42
        colorCount > 256
43
        colorCount = 10:
44
45
      if (typeof quality == "undefined" || quality < 1) {
46
        quality = 10;
47
48
49
50
      // Create custom CanvasImage object
51
      this.image = canvasImage(sourceImage);
52
      const imageData = this.image.getImageData();
53
      const pixels = imageData.data;
54
      const pixelCount = this.image.getPixelCount();
55
      this.worker.postMessage({ pixels, pixelCount, colorCount, quality });
56
57
58
    getPaletteFromUrl(imageUrl: string, quality = 10) {
59
      new Promise((resolve, reject) ⇒ {
60
        const sourceImage = new Image();
        sourceImage.crossOrigin = "Anonymous";
61
62
        sourceImage.addEventListener("load", () ⇒ {
63
          const palette = this.getPalette(sourceImage, 5, quality);
64
          resolve(( palette, imageUrl ));
66
        sourceImage.src = imageUrl;
67
        sourceImage.addEventListener("error", reject.blnd(this));
68
69
```

1 export class ColorFromImgDirective implements OnChanges

```
src: string:
 5
    @Output())
    newColor = new EventEmitter():
    worker: Worker:
    image: Partial<CanvasImage>:
 9
    private imgSrc = "";
10
    constructor() {
11
      this.worker = new Worker(new URL("./quantize.worker", import.meta.url));
12
      this.worker.onmessage = ({{ data }}) ⇒ {{ }
13
14
        this image removeCanvas():
        this newColor emit(data):
15
16
17
    ngOnChanges({ src }: SimpleChanges) {
18
      if (src.currentValue == "assets/imgs/default.svg") {
19
        if (src.firstChange) {
20
21
          this.imgSrc = src.currentValue;
22
          this.getPaletteFromUrl((this.imgSrc));
23
        else
24
          const incomingVal = new URL(src.currentValue);
          const currentVal = new URL(src.previousValue);
25
          if (currentVal pathname == incomingVal pathname) {
26
27
            this.imgSrc = src.currentValue:
28
            this.getPaletteFromUrl(this.imgSrc)
29
30
31
32
33
34
    getPalette(
35
      sourceImage: HTMLImageElement.
36
      colorCount?: number.
37
      quality?: number
38
      if (
39
        typeof colorCount === "undefined" ||
40
        colorCount < 2
41
        colorCount > 256
42
43
44
        colorCount = 10:
45
      if (typeof quality == "undefined" || quality < 1) {
46
        quality = 10:
47
48
49
      // Create custom CanvasImage object
50
51
      this.image = canvasImage(sourceImage):
      const imageData = this.image.getImageData();
52
53
      const pixels = imageData.data;
      const pixelCount = this.image.getPixelCount();
54
      this.worker.postMessage({{ pixels, pixelCount, colorCount, quality }});
55
56
57
    getPaletteFromUrl(imageUrl: string, quality = 10) (
58
      new Promise((resolve, reject) ⇒ {
59
        const sourceImage = new Image();
60
        sourceImage.crossOrigin = "Anonymous";
61
        sourceImage.addEventListener("load", () ⇒ {
62
          const palette = this.getPalette(sourceImage, 5, quality);
63
          resolve({{ palette, imageUrl }}):
64
65
66
        sourceImage.src = imageUrl?
67
        sourceImage.addEventListener("error", reject.bind(this));
68
69
70
```

1 export class ColorFromImgDirective implements OnChanges {

@Input()