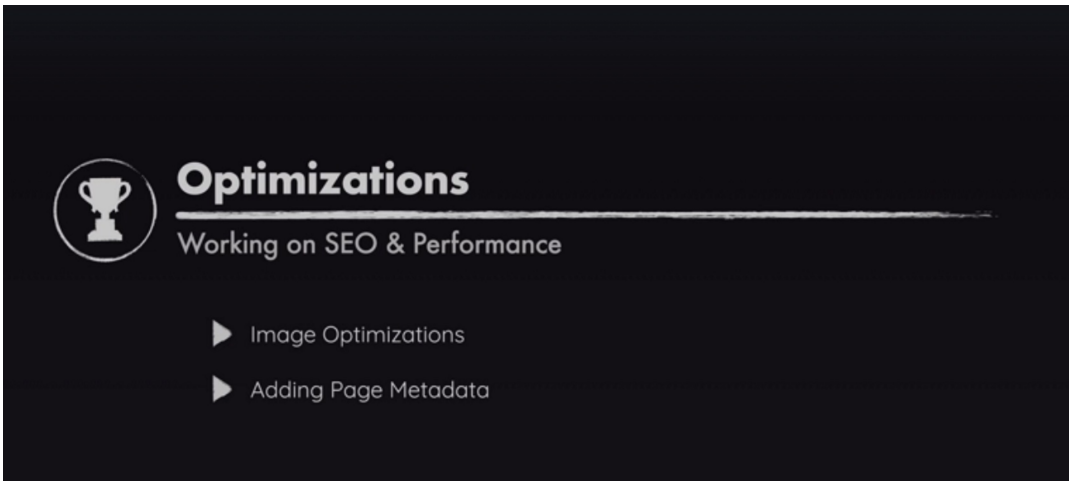
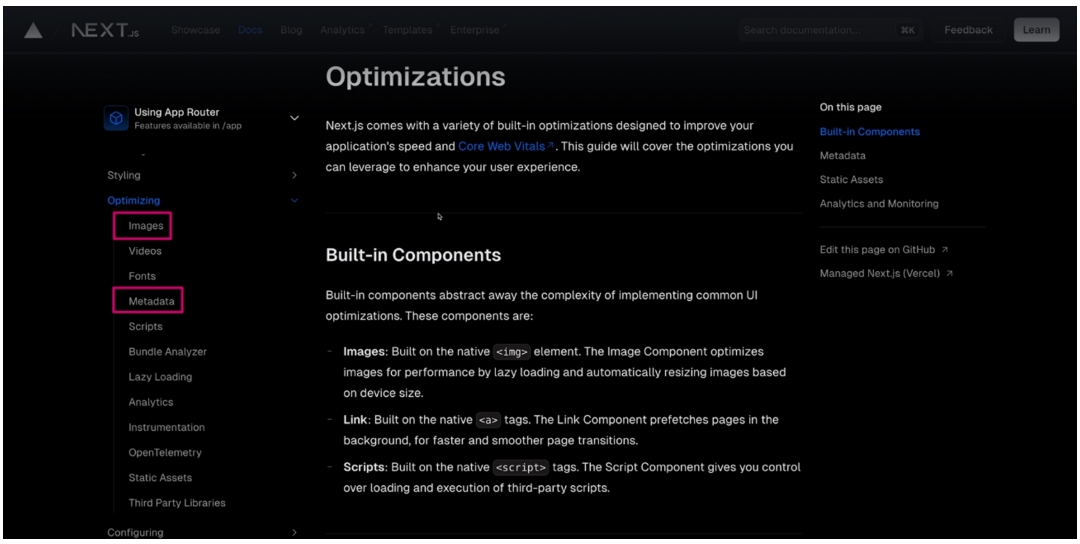
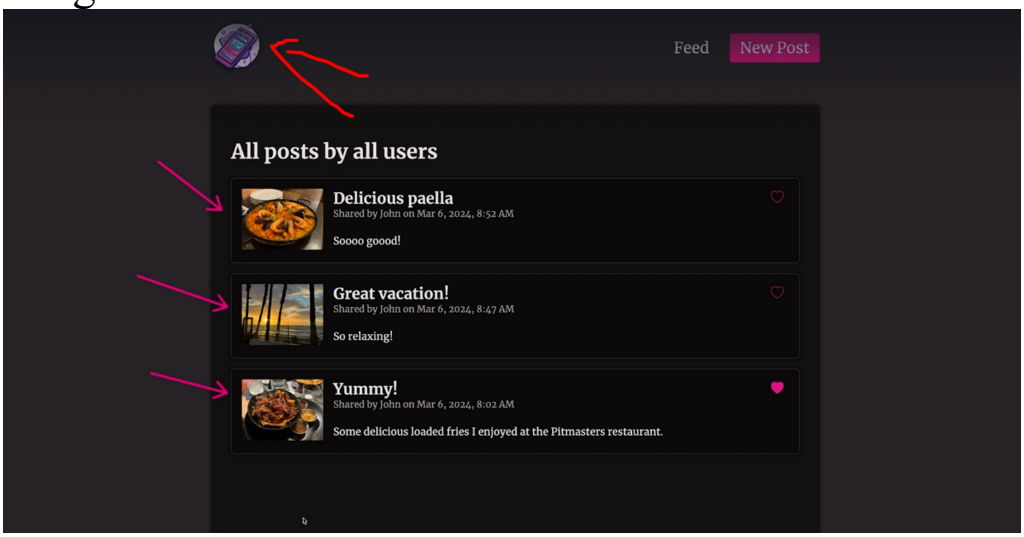
Section 8: NextJS App Optimizations

15 September 2024 10:59





In this section, we are going to optimize these images.



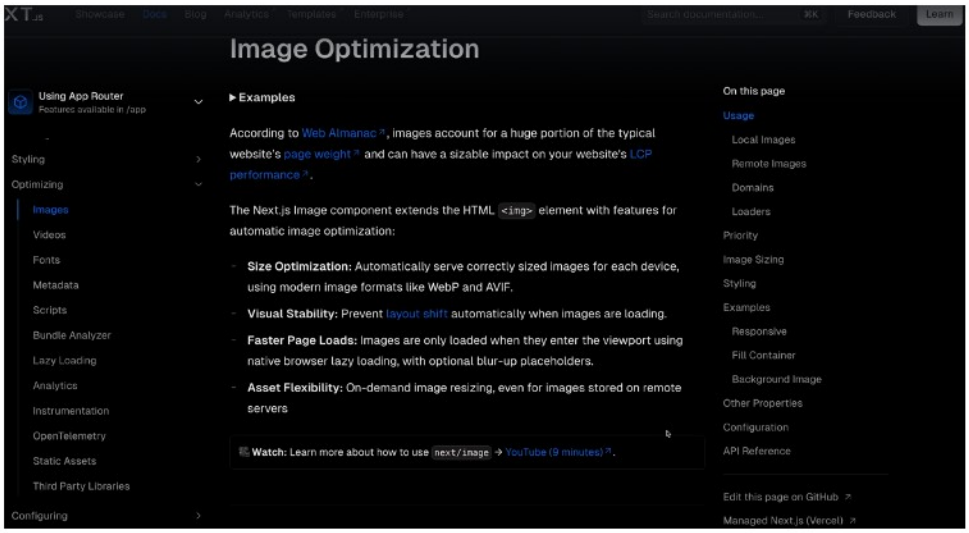
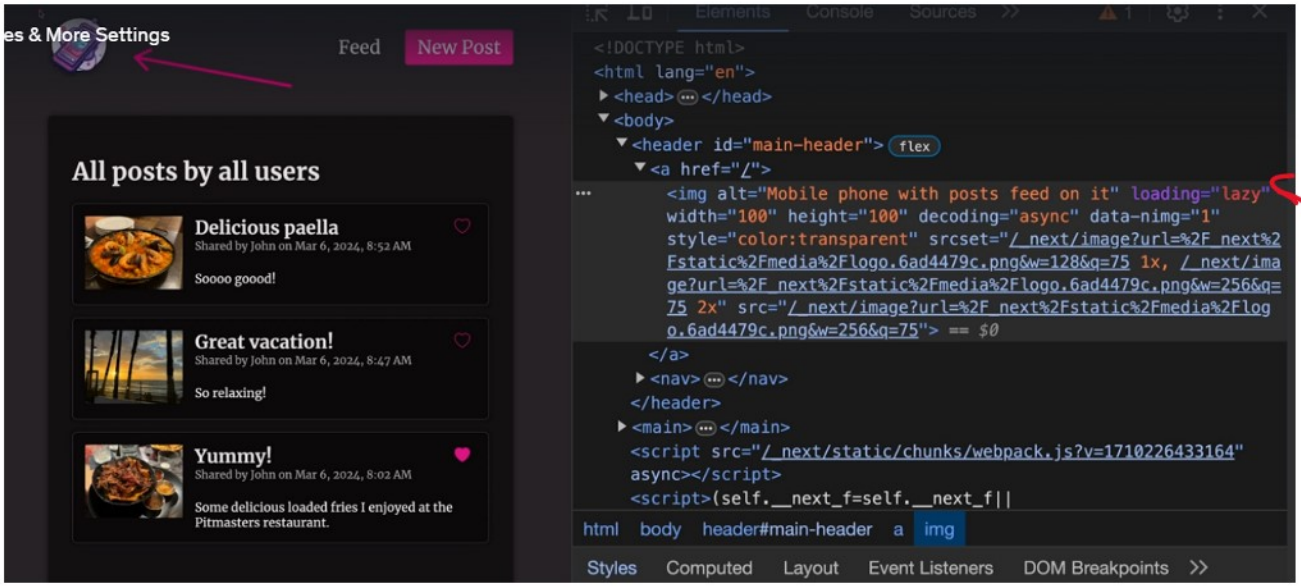


Image Optimization
Using App Router
Features available in /app
Styling
Optimizing
Images
Videos
Fonts
Metadata
Scripts
Bundle Analyzer
Lazy Loading
Analytics
Instrumentation
OpenTelemetry
Static Assets
Third Party Libraries
Configuring
▶ Examples
According to Web Almanac, images account for a huge portion of the typical website's page weight and can have a sizable impact on your website's LCP performance.
The Next.js Image component extends the HTML <img> element with features for automatic image optimization:
Size Optimization: Automatically serve correctly sized images for each device, using modern image formats like WebP and AVIF.
Visual Stability: Prevent layout shift automatically when images are loading.
Faster Page Loads: Images are only loaded when they enter the viewport using native browser lazy loading, with optional blur-up placeholders.
Asset Flexibility: On-demand image resizing, even for images stored on remote servers
Watch: Learn more about how to use next/image → YouTube (9 minutes).
On this page
Usage
Local Images
Remote Images
Domains
Loaders
Priority
Image Sizing
Styling
Examples
Responsive
Fill Container
Background Image
Other Properties
Configuration
API Reference
Edit this page on GitHub
Managed Next.js (Vercel)


es & More Settings
Feed
New Post
All posts by all users
Delicious paella
Shared by John on Mar 6, 2024, 8:52 AM
Soooo goood!
Great vacation!
Shared by John on Mar 6, 2024, 8:47 AM
So relaxing!
Yummy!
Shared by John on Mar 6, 2024, 8:02 AM
Some delicious loaded fries I enjoyed at the Pitmasters restaurant.
Elements
Console
Sources
<!DOCTYPE html>
<html lang="en">
<head>…</head>
<body>
<header id="main-header">flex

<img alt="Mobile phone with posts feed on it" loading="lazy" width="100" height="100" decoding="async" data-nimg="1" style="color:transparent" srcset="/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Flogo.6ad4479c.png&w=128&q=75 1x, /_next/image?url=%2F_next%2Fstatic%2Fmedia%2Flogo.6ad4479c.png&w=256&q=75 2x" src="/_next/image?url=%2F_next%2Fstatic%2Fmedia%2Flogo.6ad4479c.png&w=256&q=75"> == $0

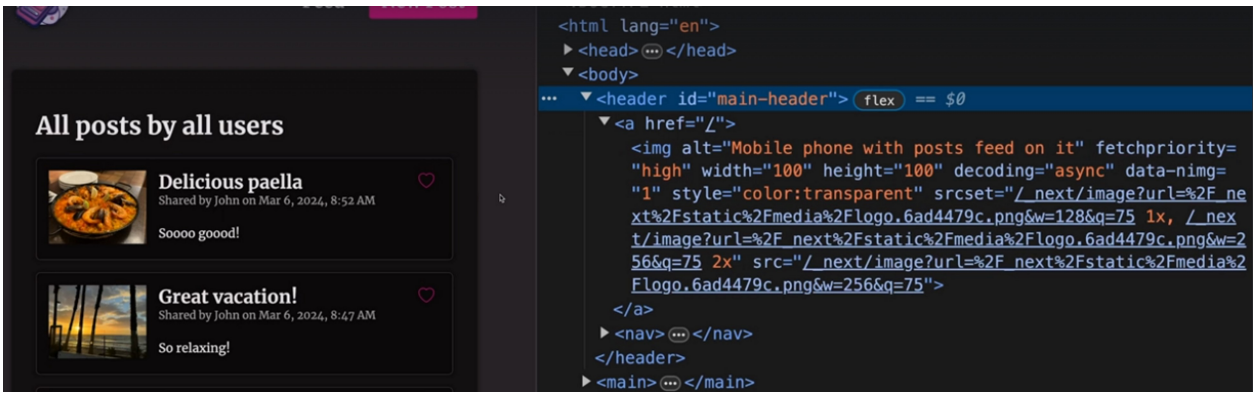
<nav>…</nav>
</header>
<main>…</main>
<script src="/_next/static/chunks/webpack.js?v=1710226433164" async></script>
<script>(self.__next_f=self.__next_f||
html body header#main-header a img
Styles
Computed
Layout
Event Listeners
DOM Breakpoints

```
export default function Header() {
  return (
    <header id="main-header">
      <Link href="/">
        <Image
          src={logo}
          width={100}
          height={100}
          priority
          alt="Mobile phone with posts feed on it"
        />
      </Link>
      <nav>
```

that's worth noting about using the image component on this image is that, as mentioned by default,
NextJS sets this loading attribute on the native image element to lazy, and that's the attribute that's supported
by all modern browsers that will make sure that the image only loaded and displayed if it's visible on the screen.

Now the problem we have here is that this image here is in the header, and therefore it will always be visible on the screen
when the page loads, so lazy loading this image here doesn't make any sense.
Instead, it's just an unnecessary check for the browser to find out whether this image needs to be loaded or not,
because we as a developer already know that it always needs to be loaded.

So therefore, what you should do if you have an image where you know with certainty that it will be displayed on the screen
when the page loads, is you should add another special prop to this image component, the priority prop.
This tells NextJS, and therefore in the end, the browser, that this image will always be visible when the page loads
and therefore when navigating to that page, NextJS would preload the image and lazy loading will also be disabled
so that the browser doesn't need to check whether the image should be loaded or not,
but that it will instead always load it. So after adding priority here and saving that file, if I reload, you see that now this lazy
loading attribute has been removed, and instead this fetchpriority, which is set to high, attribute has been added.
And therefore you want to consider adding this priority prop here to all images where you know with certainty
that they will be visible after the page has been loaded.



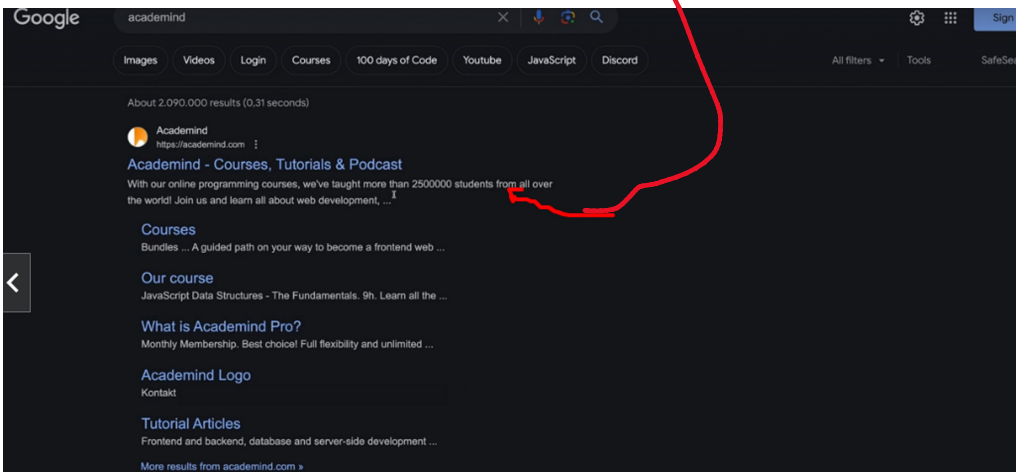
So after adding priority here and saving that file, if I reload, you see that now this lazy loading attribute has been removed, and instead this fetchpriority, which is set to high, attribute has been added.
And therefore you want to consider adding this priority prop here to all images where you know with certainty that they will be visible after the page has been loaded. That's definitely something you should consider.

Why do images that are loaded with help of the next/image and that fill prop
take up the entire available screen space? Well, because the fill prop essentially tells NextJS
that we don't know the exact size of the loaded image, and that if they offer, should fill up
all the space it can get, and by default, if you don't have any other settings, so to say,

that will be the entire screen space. Now, that's, of course, not what we want, and therefore, what you should do
when you're using the next/image with the fill prop is you should go to the container element of that image,
or add such a container element if you don't have one, and change the CSS style settings on that container.

Configuring Static Page MetaData

```
export const metadata = {
  title: 'Latest Post',
  description: ''
};
```



https://nextjs.org/docs/app/building-your-application/optimizing/metadata