

**Best Practices for Adding Google Tag to a Next.js Website**

The Google tag script you've provided needs to be properly integrated into your Next.js application to ensure optimal loading and functionality. Next.js offers specialized methods for handling third-party scripts that are more efficient than simply copying the code into every page as suggested by the standard Google instructions.

**Using Next.js Script Component (Recommended Approach)**

Since Next.js version 11, the recommended approach for implementing Google Analytics or Google Tag Manager is to use the built-in Script component from next/script. This component provides optimized script loading strategies that improve performance.

**Implementation in \_app.js**

The most efficient method is to add your Google tag in the main App component:

import Script from 'next/script';  
  
function MyApp({ Component, pageProps }) {  
 return (  
 <>  
 <Script  
 src="https://www.googletagmanager.com/gtag/js?id=AW-16691848441"  
 strategy="afterInteractive"  
 />  
 <Script id="google-tag" strategy="afterInteractive">  
 {`  
 window.dataLayer = window.dataLayer || [];  
 function gtag(){dataLayer.push(arguments);}  
 gtag('js', new Date());  
 gtag('config', 'AW-16691848441');  
 `}  
 </Script>  
 <Component {...pageProps} />  
 </>  
 );  
}  
  
export default MyApp;

The strategy="afterInteractive" parameter ensures the script loads after the page becomes interactive, which improves initial page load performance while still capturing analytics data[[1]](#fn1).

**Alternative: Implementation in \_document.js**

Another approach is to add the Google tag to the \_document.js file, which is useful for scripts that need to be loaded on every page:

import Document, { Html, Head, Main, NextScript } from "next/document";  
import Script from 'next/script';  
  
export default class MyDocument extends Document {  
 render() {  
 return (  
 <Html>  
 <Head>  
 <Script  
 src="https://www.googletagmanager.com/gtag/js?id=AW-16691848441"  
 strategy="afterInteractive"  
 />  
 </Head>  
 <body>  
 <Main />  
 <NextScript />  
 <Script id="google-tag" strategy="afterInteractive">  
 {`  
 window.dataLayer = window.dataLayer || [];  
 function gtag(){dataLayer.push(arguments);}  
 gtag('js', new Date());  
 gtag('config', 'AW-16691848441');  
 `}  
 </Script>  
 </body>  
 </Html>  
 );  
 }  
}

**Special Considerations for Google Tag Implementation**

**Production-Only Implementation**

For better development experience, you might want to load analytics scripts only in production environments:

import Script from 'next/script';  
  
function MyApp({ Component, pageProps }) {  
 const isProduction = process.env.NODE\_ENV === "production";  
   
 return (  
 <>  
 {isProduction && (  
 <>  
 <Script  
 src="https://www.googletagmanager.com/gtag/js?id=AW-16691848441"  
 strategy="afterInteractive"  
 />  
 <Script id="google-tag" strategy="afterInteractive">  
 {`  
 window.dataLayer = window.dataLayer || [];  
 function gtag(){dataLayer.push(arguments);}  
 gtag('js', new Date());  
 gtag('config', 'AW-16691848441');  
 `}  
 </Script>  
 </>  
 )}  
 <Component {...pageProps} />  
 </>  
 );  
}  
  
export default MyApp;

This approach prevents analytics from tracking during development and only enables it in production builds[[1]](#fn1).

**Page Navigation Tracking for Single-Page Applications**

Next.js applications behave like single-page applications (SPAs), meaning page changes don't trigger a full reload by default. To track page navigation properly, you should listen for route changes:

import { useEffect } from 'react';  
import { useRouter } from 'next/router';  
  
function MyApp({ Component, pageProps }) {  
 const router = useRouter();  
   
 useEffect(() => {  
 const handleRouteChange = (url) => {  
 window.gtag('config', 'AW-16691848441', {  
 page\_path: url,  
 });  
 };  
   
 router.events.on('routeChangeComplete', handleRouteChange);  
 return () => {  
 router.events.off('routeChangeComplete', handleRouteChange);  
 };  
 }, [router.events]);  
   
 // Rest of your code with Script components  
}

This ensures accurate page tracking as users navigate through your Next.js application[[1]](#fn1).

**Advanced Google Tag Capabilities**

Beyond basic page tracking, you can implement additional tracking features in your Next.js application:

**Event Tracking**

Track specific user interactions by dispatching custom events:

// Example button click tracking  
function handleButtonClick() {  
 window.gtag('event', 'button\_click', {  
 'event\_category': 'engagement',  
 'event\_label': 'sign\_up\_button'  
 });  
 // Continue with normal button functionality  
}

**User Identification and Custom Dimensions**

For more detailed analytics, implement user identification and custom attributes:

// After user logs in or is identified  
window.gtag('set', 'user\_id', 'USER\_ID');  
  
// Adding custom dimensions  
window.gtag('set', {  
 'custom\_dimension1': 'premium\_user',  
 'custom\_dimension2': 'returned\_visitor'  
});

These advanced tracking capabilities provide deeper insights into user behavior and application performance[[2]](#fn2).

**Testing Your Google Tag Implementation**

To verify your Google Tag is working correctly:

1. Install the Google Tag Assistant browser extension
2. Use the browser's developer console to check if gtag function is defined
3. Look for network requests to Google's analytics servers
4. Set up debug mode in Google Tag Manager to validate events

**Conclusion**

The best practice for implementing Google Tag in a Next.js application is to use the built-in Script component rather than directly copying the provided code snippet into every page's <head> section. This approach improves performance, ensures proper loading sequence, and works seamlessly with Next.js's routing system.

By implementing in either the App component or Document component with the appropriate strategy, you'll have a more efficient analytics implementation that follows Next.js best practices while still collecting all the data you need for your marketing and analytics purposes.

⁂

1. <https://stackoverflow.com/questions/60411351/how-to-use-google-analytics-with-next-js-app>

1. <https://www.youtube.com/watch?v=BT9uEOQ0UoY>