useReportWebVitals

The `useReportWebVitals` hook allows you to report Core Web Vitals, and can be used in combination with your analytics service.

pages/_app.js

```js
import { useReportWebVitals } from 'next/web-vitals';

function MyApp({ Component, pageProps }) {
  useReportWebVitals((metric) => {
    console.log(metric);
  });

  return <Component {...pageProps} />;
}
```

useReportWebVitals

The `metric` object passed as the hook's argument consists of a number of properties:

- `id`: Unique identifier for the metric in the context of the current page load
- `name`: The name of the performance metric. Possible values include names of Web Vitals metrics (TTFB, FCP, LCP, FID, CLS) specific to a web application.
- `delta`: The difference between the current value and the previous value of the metric. The value is typically in milliseconds and represents the change in the metric's value over time.
- `entries`: An array of Performance Entries associated with the metric. These entries provide detailed information about the performance events related to the metric.
- `navigationType`: Indicates the type of navigation that triggered the metric collection. Possible values include `"navigate"`, `"reload"`, `"back_forward"`, and `"prerender"`.

- `rating`: A qualitative rating of the metric value, providing an assessment of the performance. Possible values are `"good"`, `"needs-improvement"`, and `"poor"`. The rating is typically determined by comparing the metric value against predefined thresholds that indicate acceptable or suboptimal performance.
- `value`: The actual value or duration of the performance entry, typically in milliseconds. The value provides a quantitative measure of the performance aspect being tracked by the metric. The source of the value depends on the specific metric being measured and can come from various Performance API s.

Web Vitals

Web Vitals are a set of useful metrics that aim to capture the user experience of a web page. The following web vitals are all included:

- Time to First Byte (TTFB)
- First Contentful Paint (FCP)
- Largest Contentful Paint (LCP)
- First Input Delay (FID)
- Cumulative Layout Shift (CLS)
- Interaction to Next Paint (INP)

You can handle all the results of these metrics using the `name` property.

pages/_app.js

```js
import { useReportWebVitals } from 'next/web-vitals';

function MyApp({ Component, pageProps }) {
  useReportWebVitals((metric) => {
    switch (metric.name) {
      case 'FCP': {
        // handle FCP results
      }
      case 'LCP': {
        // handle LCP results
      }
      // ...
    }
  });

  return <Component {...pageProps} />;
}
```

Custom Metrics

In addition to the core metrics listed above, there are some additional custom metrics that measure the time it takes for the page to hydrate and render:

- `Next.js-hydration`: Length of time it takes for the page to start and finish hydrating (in ms)
- `Next.js-route-change-to-render`: Length of time it takes for a page to start rendering after a route change (in ms)
- `Next.js-render`: Length of time it takes for a page to finish render after a route change (in ms)

You can handle all the results of these metrics separately:

```
export function reportWebVitals(metric) {
  switch (metric.name) {
    case 'Next.js-hydration':
      // handle hydration results
      break;
    case 'Next.js-route-change-to-render':
      // handle route-change to render results
      break;
    case 'Next.js-render':
      // handle render results
      break;
    default:
      break;
  }
}
```

These metrics work in all browsers that support the User Timing API ↗.

Usage on Vercel

Vercel Speed Insights ↗ are automatically configured on Vercel deployments, and don't require the use of `useReportWebVitals`. This hook is useful in local development, or if you're using a different analytics service.

Sending results to external systems

You can send results to any endpoint to measure and track real user performance on your site. For example:

```js
useReportWebVitals((metric) => {
  const body = JSON.stringify(metric);
  const url = 'https://example.com/analytics';

  // Use `navigator.sendBeacon()` if available, falling back to `fetch()`.
  if (navigator.sendBeacon) {
    navigator.sendBeacon(url, body);
  } else {
    fetch(url, { body, method: 'POST', keepalive: true });
  }
});
```

Note: If you use Google Analytics ↗, using the `id` value can allow you to construct metric distributions manually (to calculate percentiles, etc.)

```js
useReportWebVitals(metric => {
  // Use `window.gtag` if you initialized Google Analytics as this example:
  // https://github.com/vercel/next.js/blob/canary/examples/with-google-analytics/page
  window.gtag('event', metric.name, {
    value: Math.round(metric.name === 'CLS' ? metric.value * 1000 : metric.value), //
    event_label: metric.id, // id unique to current page load
    non_interaction: true, // avoids affecting bounce rate.
  });
}
```

Read more about sending results to Google Analytics ↗.