Shallow Routing

▶ **Examples**

Shallow routing allows you to change the URL without running data fetching methods again, that includes `getServerSideProps`, `getStaticProps`, and `getInitialProps`.

You'll receive the updated `pathname` and the `query` via the `router` object (added by `useRouter` or `withRouter`), without losing state.

To enable shallow routing, set the `shallow` option to `true`. Consider the following example:

```
import { useEffect } from 'react'
import { useRouter } from 'next/router'

// Current URL is '/'
function Page() {
  const router = useRouter()

  useEffect(() => {
    // Always do navigations after the first render
    router.push('/?counter=10', undefined, { shallow: true })
  }, [])

  useEffect(() => {
    // The counter changed!
  }, [router.query.counter])
}

export default Page
```

The URL will get updated to `/?counter=10`. and the page won't get replaced, only the state of the route is changed.

You can also watch for URL changes via `componentDidUpdate` as shown below:

```
componentDidUpdate(prevProps) {
  const { pathname, query } = this.props.router
  // verify props have changed to avoid an infinite loop
  if (query.counter !== prevProps.router.query.counter) {
    // fetch data based on the new query
  }
}
```

Caveats

Shallow routing **only** works for URL changes in the current page. For example, let's assume we have another page called `pages/about.js`, and you run this:

```
router.push('/?counter=10', '/about?counter=10', { shallow: true })
```

Since that's a new page, it'll unload the current page, load the new one and wait for data fetching even though we asked to do shallow routing.