Custom `Document`

A custom `Document` can update the `<html>` and `<body>` tags used to render a Page. This file is only rendered on the server, so event handlers like `onClick` cannot be used in `_document`.

To override the default `Document`, create the file `pages/_document.js` as shown below:

```
import { Html, Head, Main, NextScript } from 'next/document'

export default function Document() {
  return (
    <Html>
      <Head />
      <body>
        <Main />
        <NextScript />
      </body>
    </Html>
  )
}
```

The code above is the default `Document` added by Next.js. Custom attributes are allowed as props. For example, we might want to add `lang="en"` to the `<html>` tag:

```
<Html lang="en">
```

Or add a `className` to the `body` tag:

```
<body className="bg-white">
```

`<Html>`, `<Head />`, `<Main />` and `<NextScript />` are required for the page to be properly rendered.

Caveats

- The `<Head />` component used in `_document` is not the same as `next/head`. The `<Head />` component used here should only be used for any `<head>` code that is common for all pages. For all other cases, such as `<title>` tags, we recommend using `next/head` in your pages or components.
- React components outside of `<Main />` will not be initialized by the browser. Do *not* add application logic here or custom CSS (like `styled-jsx`). If you need shared components in all your pages (like a menu or a toolbar), read Layouts instead.
- `Document` currently does not support Next.js Data Fetching methods like `getStaticProps` or `getServerSideProps`.

Customizing `renderPage`

> ***Note:*** *This is advanced and only needed for libraries like CSS-in-JS to support server-side rendering. This is not needed for built-in `styled-jsx` support.*

To prepare for React 18, we recommend avoiding customizing `getInitialProps` and `renderPage`, if possible.

The `ctx` object shown below is equivalent to the one received in `getInitialProps`, with the addition of `renderPage`.

```
import Document, { Html, Head, Main, NextScript } from 'next/document'

class MyDocument extends Document {
  static async getInitialProps(ctx) {
    const originalRenderPage = ctx.renderPage

    // Run the React rendering logic synchronously
    ctx.renderPage = () =>
      originalRenderPage({
        // Useful for wrapping the whole react tree
        enhanceApp: (App) => App,
        // Useful for wrapping in a per-page basis
        enhanceComponent: (Component) => Component,
      })

    // Run the parent `getInitialProps`, it now includes the custom `renderPage`
    const initialProps = await Document.getInitialProps(ctx)

    return initialProps
  }

  render() {
    return (
      <Html>
        <Head />
        <body>
          <Main />
          <NextScript />
        </body>
      </Html>
    )
  }
}

export default MyDocument
```

> ***Note**: `getInitialProps` in `_document` is not called during client-side transitions.*

TypeScript

You can use the built-in `DocumentContext` type and change the file name to `./pages/_document.tsx` like so:

```
import Document, { DocumentContext } from 'next/document'

class MyDocument extends Document {
  static async getInitialProps(ctx: DocumentContext) {
    const initialProps = await Document.getInitialProps(ctx)

    return initialProps
  }
}

export default MyDocument
```