Toggly Remix

Loaders · Actions · SSR hydration · React hooks

Install

Use Remix 2 and React 18. React 19 and the React Router framework starter are outside the client peer range.

```
npm i @ops-ai/remix-toggly-core \
  @ops-ai/remix-toggly-server \
  @ops-ai/remix-toggly-client
```

Requires Remix 2.x, React 18.x, Node.js 18+.

Loader helper

APP/UTILS/TOGGLY.SERVER.TS

```
import { createTogglyLoader } from
  '@ops-ai/remix-toggly-server';

export const togglyLoader = createTogglyLoader({
  appKey: process.env.TOGGLY_APP_KEY,
  environment: process.env.TOGGLY_ENVIRONMENT,
  getIdentity: async (request) => {
    const session = await getSession(
      request.headers.get('Cookie'));
    return session.get('userId');
  },
});
```

Root loader

```
// app/root.tsx
export async function loader({ request }) {
  return togglyLoader.getLoaderData({
    request, params: {}, context: {} });
}

export default function App() {
  return (
    <html>
      <body>
        <RemixTogglyProvider>
          <Outlet />
        </RemixTogglyProvider>
        <Scripts />
      </body>
    </html>
  );
}
```

Loaders & actions

REQUEST-BOUND LOADER

```
export async function loader(args) {
  return togglyLoader.run(args, async () => {
    const on = await togglyLoader.isEnabled(
      'new-api-v2');
    return json(on
      ? await fetchNewAPI()
      : await fetchOldAPI());
  });
}
```

INSIDE THE RUN CALLBACK

```
const hasAccess =
  await togglyLoader.evaluateGate(
    ['admin-panel', 'super-user'], 'any');
if (!hasAccess) {
  throw redirect('/unauthorized');
}
```

ACTION GATING

```
export async function action(args) {
  return togglyLoader.run(args, async () => {
    const on = await togglyLoader.isEnabled(
      'new-form');
    if (!on) throw new Response('Disabled', {
      status: 403 });
    // process form...
  });
}
```

Loader + extra data

```
export async function loader({ request }) {
  const products = await getProducts();
  return togglyLoader.getLoaderData(
    { request, params: {}, context: {} },
    { products },
  );
}
```

Toggly Remix

Loaders · Actions · SSR hydration · React hooks

Client provider

```
import { RemixTogglyProvider } from
  '@ops-ai/remix-toggly-client';

<RemixTogglyProvider
  enableRefresh
  refreshInterval={30000}
  onFlagsChange={(flags) => …}>
  <Outlet />
</RemixTogglyProvider>
```

Hydrates automatically from loader data — no manual flag passing needed.

Hooks

```
import { useFeature, useFeatureGate,
  useFeatures, useToggly } from
  '@ops-ai/remix-toggly-client';

const isOn = useFeature('new-ui');
const isEnabled = useFeatureGate(
  ['a','b'], 'any');
const { flags, refresh, identify }
  = useToggly();
```

Components

```
import { Feature } from
  '@ops-ai/remix-toggly-client';

<Feature featureKey="new-analytics">
  <AdvancedAnalytics />
</Feature>
<Feature featureKey="new-analytics" negate>
  <BasicAnalytics />
</Feature>

{isOn && <BetaFeatures />}
```

Identity

```
const { identify, reset } = useToggly();

// on login
await identify(user.id);

// on logout
await reset();
```

Pitfalls

- Use `togglyLoader.run(args, fn)` for request-bound checks; `getLoaderData()` hydrates clients.
- Wrap the app in `RemixTogglyProvider` so client hooks hydrate from loader data.
- Use `getIdentity` in the loader helper for consistent % rollouts across SSR and client.
- Store `TOGGLY_APP_KEY` in server env vars only — never bundle it into client code.
- Use `identify()` or `reset()` to update client targeting.

Resources

- Docs: `docs.toggly.io/sdks/remix`
- Dashboard: `app.toggly.io`