Toggly JavaScript / TypeScript

Vanilla JS, React, Angular, Vue, Svelte

Install

BROWSER BUNDLE

```
npm i @ops-ai/feature-flags-toggly
```

FRAMEWORK WRAPPERS

```
npm i @ops-ai/react-feature-flags-toggly
npm i @ops-ai/ngx-feature-flags-toggly
npm i @ops-ai/vue-feature-flags-toggly
npm i @ops-ai/svelte-feature-flags-toggly
```

CDN

```
<script src="https://cdn.jsdelivr.net/npm/
  @ops-ai/feature-flags-toggly/dist/
  feature-flags-toggly.bundle.js"></script>
```

Initialize

Returns a Promise resolving with the loaded flag map.

```
await Toggly.init({
  identity: 'user-123', // stable ID
  groups: ['beta'], // memberships
  claims: { plan: 'pro' }, // string attributes
  appKey: APP_KEY,
  environment: 'Production',
  flagDefaults: { MyFlag: false }
}); // ready to evaluate this user
```

`identity` auto-generates a UUID in `localStorage` if absent; shared across apps on that origin, not per tab. Pass known context to init; use setContext for later changes. Context stays in memory when storage is unavailable.

Boolean evaluation

```
Toggly.isFeatureOn('SignUp')   // true/false
Toggly.isFeatureOff('OldUI')
```

MULTI-FLAG GATE

```
// 0 = all, 1 = any (FeatureRequirement enum)
Toggly.evaluateFeatureGate(
  ['Beta', 'Pricing'],
  1,   // any
  false);
```

PREFER API OVER RAW MAP

```
// Use isFeatureOn — gate objects in
// featureFlagsValue are not plain booleans
Toggly.isFeatureOn('MyFlag');
```

Identity & rollouts

Set a stable `identity` for % rollouts and sticky variants.

```
await Toggly.setContext({ identity: 'user-123' });
const id = Toggly.identity;
// Direct identity assignment does not refresh flags.
```

Refresh & cache

```
Toggly.refresh();                // pull now
Toggly.startRefreshInterval();   // resume
Toggly.cancelRefreshInterval(); // pause

Toggly.cacheFeatureFlags({ A: true });
Toggly.clearFeatureFlagsCache();
```

Definitions cached in `localStorage`; default polling interval is 3 min. Server-side per-user cache: 30 min sliding.

Lifecycle hooks

Tap `beforeEvaluation`, `afterEvaluation`, `beforeIdentify`, `afterIdentify`, `afterRefresh`.

```
const ga4Hook = {
  getMetadata: () ⇒
    ({ name: 'GA4', version: '1' }),
  afterEvaluation: async d ⇒
    gtag('event', 'feature_flag_evaluated', {
      feature_name:    d.featureKey,
      feature_enabled: d.result,
    })
};
Toggly.init({ appKey, environment,
              hooks: [ga4Hook] });
Toggly.addHook(myHook);
Toggly.removeHook(myHook.getMetadata().name);
```

Pre-built: `@ops-ai/toggly-clarity-hook`, `@ops-ai/toggly-ga4-hook`, `@ops-ai/toggly-appinsights-hook`.

Browser runtime

The browser bundle exposes window.Toggly. See the sample for an app walkthrough. HTTP user-agent/country overrides are not per-call browser APIs.

Toggly JavaScript / TypeScript

Vanilla JS, React, Angular, Vue, Svelte

React

React 18.2+ and React 19: install matching React/React DOM peers. Existing React 18 setup is supported. Start with the React sample for a guided app walkthrough.

```
import { createRoot } from 'react-dom/client';
import { createTogglyProvider,
         Feature, useFeatureFlag, useFeatureGate }
  from '@ops-ai/react-feature-flags-toggly';

const TogglyProvider =
  await createTogglyProvider({
    appKey, environment, identity });

const root = createRoot(document.getElementById('root
root.render(
  <TogglyProvider><App/></TogglyProvider>);
```

COMPONENT API

```
<Feature featureKey="SignUp">…</Feature>
<Feature featureKeys={['A','B']}
         requirement="any" />
<Feature featureKeys={['A']}
         negate>…</Feature>
```

CONDITIONAL UI (RENDER PROP)

```
<Feature featureKey="PremiumCheckout"
  render={(enabled) => (
    <button className={enabled ? 'active' : ''}
            disabled={!enabled}>
      Checkout
    </button>
  )} />
```

HOOKS

```
const { isEnabled, isLoading } =
  useFeatureFlag('SignUp');
const { isEnabled } = useFeatureGate(['A','B'], {
  requirement: 'any' });
```

React Native

React Native and Expo have a dedicated cheat sheet — see `docs.toggly.io/cheatsheets/react-native`.

SolidJS

SolidJS and SolidStart have a dedicated cheat sheet — see `docs.toggly.io/cheatsheets/solid`.

Angular

```
NgxFeatureFlagsTogglyModule.forRoot({
  appKey, environment, identity })

// show/hide directive
<div *featureFlag="'SignUp'">…</div>
<div *featureFlag="['A','B']"
     featureFlagRequirement="any">…</div>

// conditional UI — template stays mounted
<button
  *featureGateBuilder="'PremiumCheckout'; let enabled
  [class.active]="enabled">
  Checkout
</button>

// service
const on = await this._toggly
                    .isFeatureOn('SignUp');
```

Vue & Svelte

VUE SHOW/HIDE

```
app.use(toggly, { appKey, environment });
<Feature feature-key="SignUp"/>
```

VUE CONDITIONAL UI

```
<FeatureGateBuilder
  feature-key="PremiumCheckout"
  v-slot="{ enabled }">
  <button :class="{ active: enabled }">
    Checkout
  </button>
</FeatureGateBuilder>

const { isEnabled } = useFeatureGate({
  featureKeys: ['PremiumCheckout'] });
```

SVELTE

```
await createToggly({ appKey, environment });
// then in components:
<Feature featureKey="SignUp">…</Feature>
<FeatureGateBuilder featureKey="PremiumCheckout"
                    let:enabled>
  <button class:active={enabled}>Checkout</button>
</FeatureGateBuilder>
```

Toggly JavaScript / TypeScript

Vanilla JS, React, Angular, Vue, Svelte

SSR frameworks

Next.js, Remix, Nuxt, Astro, Gatsby, SvelteKit, and SolidStart each have dedicated cheat sheets with server-side and client-side patterns. See `docs.toggly.io/cheatsheets`.

Pitfalls

- Call `Toggly.init` once per app — calling it again resets in-memory state and re-fetches definitions.
- Use the **App Key** in client-side code only. Never expose the management API key.
- Always provide `flagDefaults` so an offline or blocked CDN doesn't break the page.
- Set `identity` before evaluating flags — anonymous users land in the default branch of % rollouts.
- Set `featureFlagsRefreshInterval: 0` to disable polling in tests.

HTML / CSS only (no JS)

```
<link rel="stylesheet"
  href="https://client.toggly.io/
    APP_KEY-ENV/defs.css"/>

<div feature-key="Feature1 Feature2"
     feature-condition="any">
  Visible when any key is on.
</div>
```

Resources

- Docs: `docs.toggly.io/sdks/javascript`
- Cheat sheets: `docs.toggly.io/cheatsheets`
- npm: `@ops-ai/feature-flags-toggly`
- GitHub: `github.com/ops-ai`
- Dashboard: `app.toggly.io`