Remix presentation beta.gouv.fr

C'est quoi Remix ?

React Router v6

```
<Routes>
  <Route element={<RootRoute />}>
    <Route index element={<IndexRoute />} />
    <Route path="signin" element={<SignInRoute />} />
    <Route path="signout" element={<SignOutRoute />} />
    <Route path="todos" element={<TodosRoute />}>
      <Route index element={<TodosIndexRoute />} />
      <Route path="$id" element={<TodosIdRoute />} />
    </Route>
  </Route>
</Routes>
```

React Router v6 + Remix

```
<Routes>
  <Route file="root.tsx">
    <Route index file="routes/index.tsx" />
    <Route path="signin" file="routes/signin.tsx" />
    <Route path="signout" file="routes/signout.tsx" />
    <Route path="todos" file="routes/todos.tsx">
      <Route index file="routes/todos/index.tsx" />
      <Route path="$id" file="routes/todos/$id.tsx" />
    </Route>
  </Route>
</Routes>
```

Backend

RouteModule

```
export interface RouteModule {
  handle?: RouteHandle;
  links?: LinksFunction;
  meta?: MetaFunction | HtmlMetaDescriptor;
  headers?: HeadersFunction | { [name: string]: string };

  CatchBoundary?: CatchBoundaryComponent;
  ErrorBoundary?: ErrorBoundaryComponent;

  loader?: LoaderFunction;
  action?: ActionFunction;
  default: RouteComponent;
}
```

Links

```
export const links = () => [
  {
    rel: 'stylesheet',
    href: '/tailwind.css'
  },
  {
    rel: 'apple-touch-icon',
    sizes: '180x180',
    href: '/apple-touch-icon.png',
  },
  {
    rel: 'icon',
    type: 'image/png',
    sizes: '32x32',
    href: '/favicon-32x32.png',
  }
];
```

Meta

```
export const meta = ({ data }) => ({
  title: `My Beautiful Website | ${data.title}`,
  description: 'A good description'
});
```

Fetch API

```
const response = await fetch('https://my.api.com', {
  headers: { accept: 'application/json' }
});

response // type Response

const request = Request('https://my.api.com', {
  headers: { accept: 'application/json' }
});

request // type Request

await fetch(request);
```

Data Function

```
({ request, params }) => Promise<Response>;
```

- **Loader Function**: *GET* requests
- **Action Function**: *POST | PUT | PATCH | DELETE* requests

Loader

```
export const loader: LoaderFunction = async ({ params }) => {
  const todo = await getTodo(params.id);

  return new Response(JSON.stringify(todo), {
    headers: { 'content-type': 'application/json' }
  })
};
```

Loader + json

```
import { json } from 'remix';

export const loader: LoaderFunction = async ({ params }) => {
  const todo = await getTodo(params.id);

  return json(todo);
};
```

```
export const loader: LoaderFunction = async ({ params }) => {
  const todo = await getTodo(params.id);

  return todo;
};
```

```
export const loader: LoaderFunction = async ({ params }) => {
  return getTodo(params.id);
};
```

Loader + query params

```
import { json } from 'remix';

export const loader: LoaderFunction = async ({ request }) => {
  const url = new URL(request.url);
  const term = url.searchParams.get('term');

  const todos = await searchTodos(term);

  return json(todos);
};
```

Loader + fetch

```
export const loader: LoaderFunction = async ({ request }) => {
  return fetch('https://my.api.com', {
    headers: request.headers
  });
};
```

Action

```
import { redirect } from 'remix';

export const action: LoaderFunction = async ({ request, params }) => {
  const body = await request.formData();
  const firstName = body.get('firstName');

  await updateProfile(params.id, { firstName });

  return redirect(`/profile/${id}`);
};
```

Action + json

```
import { json } from 'remix';

export const action: LoaderFunction = async ({ request }) => {
  const body = await request.json();
  const { firstName } = body;

  const profile = await createProfile({ firstName });

  return json(profile, {
    status: 201,
    headers: { location: `/profile/${profile.id}` }
  });
};
```

Frontend

Server Side Rendering

```
import { renderToString } from 'react-dom/server';

function handleRequest(
  request,
  responseStatusCode,
  responseHeaders
) {
  const markup = renderToString(<RemixServer url={request.url} />);

  responseHeaders.set('content-type', 'text/html');

  return new Response('<!DOCTYPE html>' + markup, {
    status: responseStatusCode,
    headers: responseHeaders,
  });
}
```

Client Side Rendering

```
import { hydrate } from 'react-dom';

hydrate(<RemixBrowser />, document);
```

Document

```
function Document({ children }) {
  const includeScripts = useMatches()
    .some(({ handle }) => handle?.hydrate);

  return (
    <html lang="en">
      <head>
        <Meta />
        <Links />
      </head>
      <body>
        {children}
        <ScrollRestoration />
        {includeScripts ? <Scripts /> : null}
        <LiveReload />
      </body>
    </html>
  );
}
```

Application Root

```
export default function App() {
  return (
    <Document>
      <Outlet />
    </Document>
  );
}
```

ErrorBoundary

```
export function ErrorBoundary({ error }) {
  return <Document>
    <h1>500</h1>
    <p>{error.message}</p>
  </Document>;
}
```

CatchBoundary

```
export function CatchBoundary() {
  const caught = useCatch();

  return <Document>
    <h1>{caught.status}</h1>
    <p>{caught.statusText}</p>
  </Document>;
}
```

Route

```
export default IndexRoute() {
  return <div>Hello World!</div>;
}
```

⚠️ Si `export default` est omis, la route sera interprétée comme une API route !

Route + loader

```
export const loader = () => ({ message: 'Hello World!' });

export default IndexRoute() {
  const data = useLoaderData();

  return <div>{data.message}</div>;
}
```

Handle

```
export const handle = { hydrate: true };
```

useMatches

```
const includeScripts = useMatches()
  .some(({ handle }) => handle?.hydrate);
```

Mutations

HTML Form

```
function MyForm({ firstName }) {
  return <form action="/profile">
    <label htmlFor="firstName">First Name</label>
    <input
      id="firstName"
      name="firstName"
      defaultValue={firstName}
    />
    <button type="submit">Save</button>
  </form>;
}
```

Remix Form

```
function MyForm({ firstName }) {
  return <Form action="/profile" replace>
    <label htmlFor="firstName">First Name</label>
    <input
      id="firstName"
      name="firstName"
      defaultValue={firstName}
    />
    <button type="submit">Save</button>
  </Form>;
}
```

Remix Optimistic Form

```
function MyForm({ firstName }) {
  const transition = useTransition();
  const disabled = transition.state == 'submitting';
  const optimisticFirstName = transition
    .submission?.formData.get('firstName');

  return <>
    <p>{optimisticFirstName ?? firstName}</p>
    <Form action="/profile" replace>
      <label htmlFor="firstName">First Name</label>
      <input
        id="firstName"
        name="firstName"
        defaultValue={firstName}
      />
      <button disabled={disabled} type="submit">Save</button>
    </Form>
  </>;
}
```