React Router v5 vs v6 vs v7 — Code & Implementation Deep Dive

This document provides concrete, side-by-side code comparisons across React Router v5, v6, and v7 to illustrate the technical evolution of routing, data loading, navigation, parameters handling, and architectural responsibilities.

1. Route Definitions & Matching

v5 — Order-based matching with Switch

```
<Switch>
  <Route path="/users/new" component={NewUser} />
  <Route path="/users/:id" component={User} />
</Switch>
```

v6 — Route ranking removes ordering concerns

```
<Routes>
  <Route path="/users/new" element={<NewUser />} />
  <Route path="/users/:id" element={<User />} />
</Routes>
```

v7 — Same ranking logic, now with data awareness

```
createBrowserRouter([
  { path: "/users/new", element: <NewUser /> },
  { path: "/users/:id", element: <User />, loader: userLoader },
])
```

2. URL Parameters & Search Params

v5 — Params + query parsing manually

```
const { id } = useParams();
const query = new URLSearchParams(location.search);
const page = query.get("page");
```

v6 — useSearchParams helper

```
const { id } = useParams();
const [searchParams] = useSearchParams();
const page = searchParams.get("page");
```

v7 — Params available in loader before render

```
export async function loader({ params, request }) {
  const url = new URL(request.url);
  return fetchUser(params.id, url.searchParams.get("page"));
}
```

3. Data Fetching Lifecycle

v5 — useEffect-driven fetching

```
useEffect(() => {
  fetchUser(id).then(setUser);
}, [id]);
```

v6 — Same mental model, improved composition

```
useEffect(() => {
  fetchUser(id).then(setUser);
}, [id]);
```

v7 — loader executes before render

```
export async function loader({ params }) {
  return fetchUser(params.id);
}
```

4. Error Handling & Boundaries

v5/v6 — Error state inside component

```
const [error, setError] = useState(null);
if (error) return <ErrorUI />;
```

v7 — Route-level error boundaries

```
{
  path: "/users/:id",
  loader: userLoader,
  element: <User />,
  errorElement: <UserErrorBoundary />
}
```

5. Navigation & Redirects

v5 — useHistory imperative navigation
```
const history = useHistory();
history.push("/dashboard");
```

v6 — useNavigate hook
```
const navigate = useNavigate();
navigate("/dashboard");
```

v7 — Redirects inside loaders/actions
```
if (!isLoggedIn()) throw redirect("/login");
```

6. Forms, Mutations & Actions

v5/v6 — Manual submit + fetch

```
const onSubmit = async (e) => {
  e.preventDefault();
  await saveData();
};
```

v7 — Declarative Form + action

```
<Form method="post">
  <button type="submit">Save</button>
</Form>
```

7. Loading & Navigation State

v5/v6 — Local loading flags

```
const [loading, setLoading] = useState(true);
```

v7 — Global navigation state

```
const navigation = useNavigation();
if (navigation.state !== "idle") showSpinner();
```

8. Architectural Responsibility Shift

v5 and v6 treat routing primarily as a UI concern. v7 elevates routing to a first-class application orchestration layer, coordinating URL, data, errors, and mutations.