Next.js Link examples

1. Link components, use of 'replace' inside key Link
2. Give different type of example (like dynamic link, dynamic link using props, click, etc)
For the Next.js 15 in typescripts (.tsx)

Here's a clear and practical explanation on how to use `<Link>` in **Next.js 15** with **TypeScript (`.tsx`)**, especially around the `replace` prop, dynamic routing, and different use cases like props and click handlers.

✅ 1. Basic Link with `replace` prop

In Next.js, the `replace` prop on a `<Link>` component prevents adding a new entry in the browser's history stack. Instead, it replaces the current entry.

```tsx
// pages/index.tsx
import Link from 'next/link';

export default function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      <Link href="/about" replace>
        <button>Go to About (Replace history)</button>
      </Link>
    </div>
  );
}
```

🔹 When the user clicks this link, they will go to `/about`, but pressing the back button won't return them to `/`.

✅ 2. Dynamic Link (using dynamic route)

Suppose you have a dynamic route like this:

```bash
/pages/post/[id].tsx
```

Then you can navigate to a post dynamically like this:

```tsx
// pages/index.tsx
import Link from 'next/link';

export default function Home() {
  return (
    <div>
      <h1>Posts</h1>
      <Link href="/post/123">Go to Post 123</Link>
    </div>
  );
}
```

And the dynamic route:

```tsx
// pages/post/[id].tsx
import { useRouter } from 'next/router';

export default function PostPage() {
  const router = useRouter();
  const { id } = router.query;

  return <div>Post ID: {id}</div>;
}
```

✅ 3. Dynamic Link Using Props

You can make a reusable component that takes a dynamic route via props:

```tsx
// components/PostLink.tsx
import Link from 'next/link';
```

```tsx
interface PostLinkProps {
  postId: string;
}

export const PostLink: React.FC<PostLinkProps> = ({ postId }) => {
  return <Link href={`/post/${postId}`}>View Post {postId}</Link>;
};
```

Usage:

```tsx
// pages/index.tsx
import { PostLink } from '@/components/PostLink';

export default function Home() {
  return (
    <div>
      <PostLink postId="456" />
    </div>
  );
}
```

✅ 4. Link navigation via click (using `router.push`)

Sometimes, you might want to navigate programmatically on a button click.

```tsx
// pages/index.tsx
'use client';

import { useRouter } from 'next/navigation';

export default function Home() {
  const router = useRouter();

  const handleClick = () => {
    router.push('/about');
    // Or: router.replace('/about'); // for replacing history
```

```
  };

  return <button onClick={handleClick}>Go to About Page</button>;
}
```

🔁 Summary Table

Use Case	Code Snippet
Basic link	`<Link href="/about">About</Link>`
With `replace`	`<Link href="/about" replace>About</Link>`
Dynamic route	`<Link href="/post/123">Post 123</Link>`
Dynamic via props	`<PostLink postId="456" />`
Programmatic onClick navigation	`router.push('/about')`

Let me know if you want examples using `searchParams` or `query strings` too.

How does 'replace' exactly works?

Great question.

🔁 How `replace` works in Next.js `<Link>` or `router.replace`

✅ What is `replace`?

In Next.js (and also in browsers in general), navigation usually **adds a new entry** to the browser's **history stack**. This means that when users press the **Back** button, they can return to the previous page.

But when you use the `replace` prop (or `router.replace()`), it **replaces the current entry** in the history instead of adding a new one.

📌 Example to Understand

```tsx
<Link href="/about" replace>
  Go to About
</Link>
```

or

```tsx
router.replace('/about');
```

⚙️ What happens:

- Current URL: `/home`
- You click the link to `/about` with `replace`
- Now you're on `/about`
- BUT: The browser **replaces** the `/home` entry with `/about`
- So if you now press the Back button, you **won't** go back to `/home`, because it's gone from the history stack

---

🧠 Browser History Analogy

Imagine browser history as a stack (like a deck of cards):

Action	History Stack
Start at `/home`	`/home`
Click Link to `/about`	`/home` → `/about`

Action	History Stack
Click Back	✅ Back to `/home`

Now with `replace`:

Action	History Stack
Start at `/home`	`/home`
Click Link to `/about` (replace)	`/about` (replaces `/home`)
Click Back	❌ Nothing happens (no `/home` anymore)

💡 When to Use `replace`

Use `replace` when:

- You **don't want** the user to go back to the previous page (e.g., after login).
- You're doing redirects and want to keep history clean.
- You're stepping through a multi-step process and want to prevent backtracking.

✅ Examples

tsx

```tsx
<Link href="/dashboard" replace>
  Go to Dashboard
</Link>
```

tsx

```tsx
const router = useRouter();
router.replace('/success');
```

Let me know if you want to see this in action or combine `replace` with `query params` or search filters.