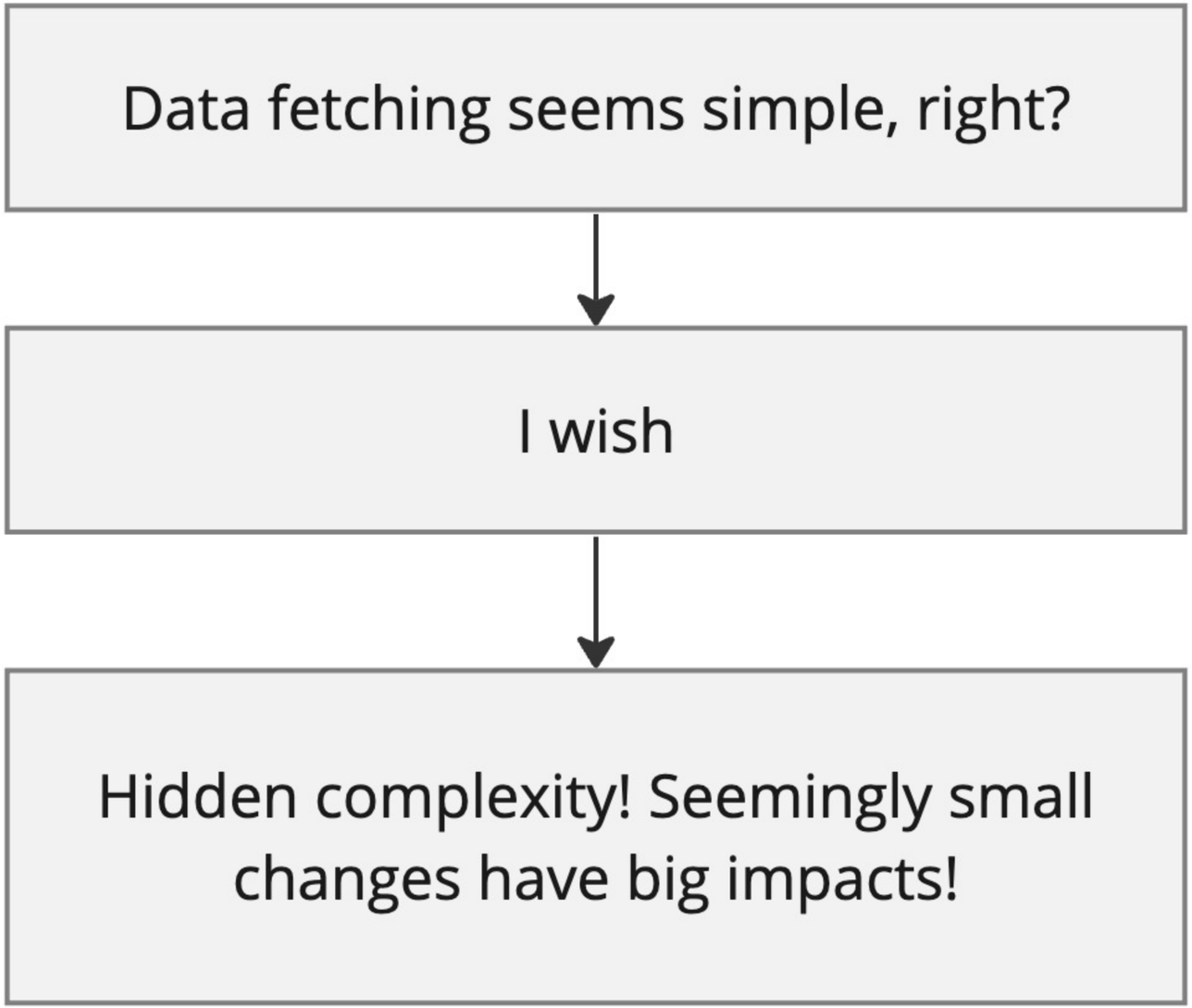
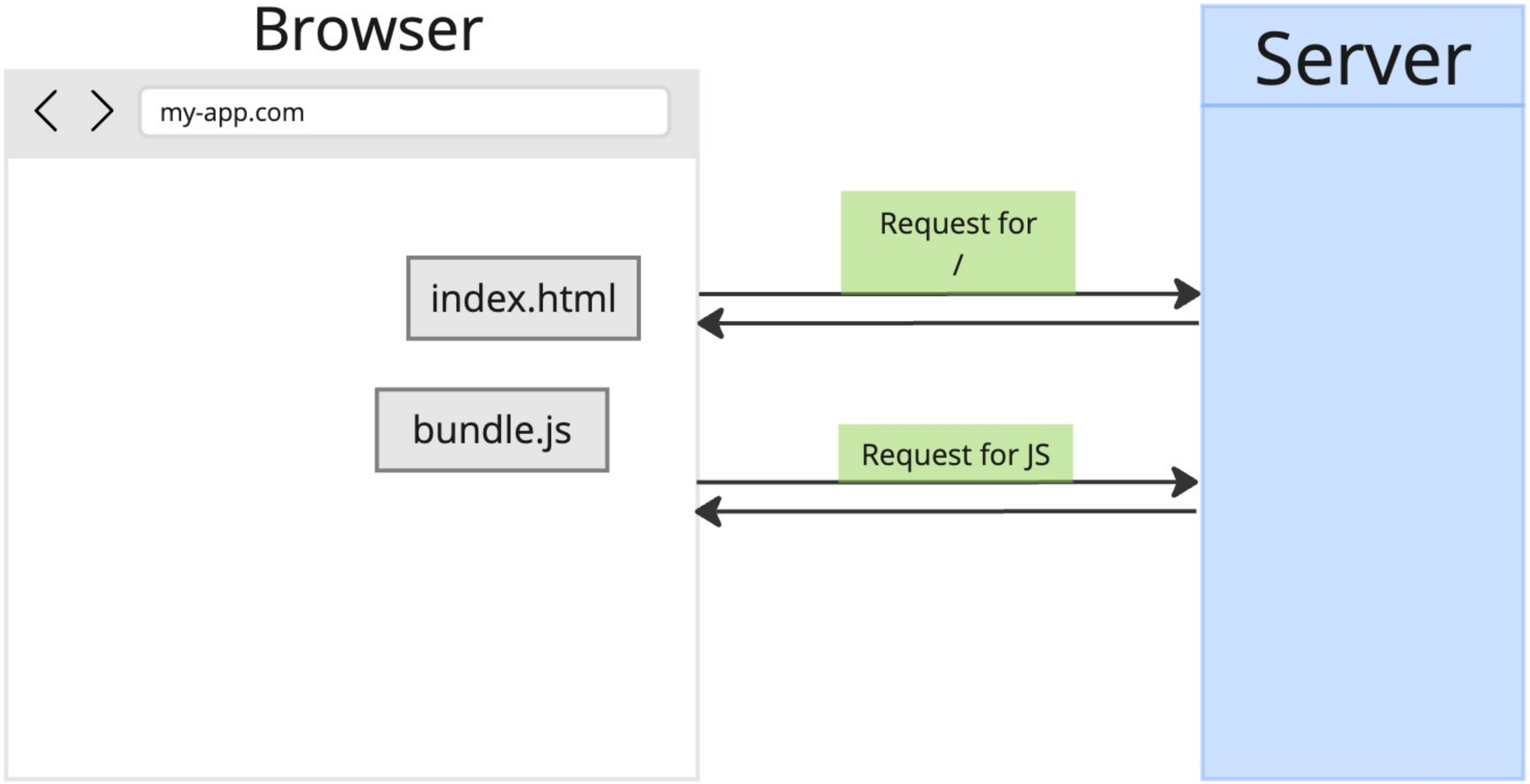
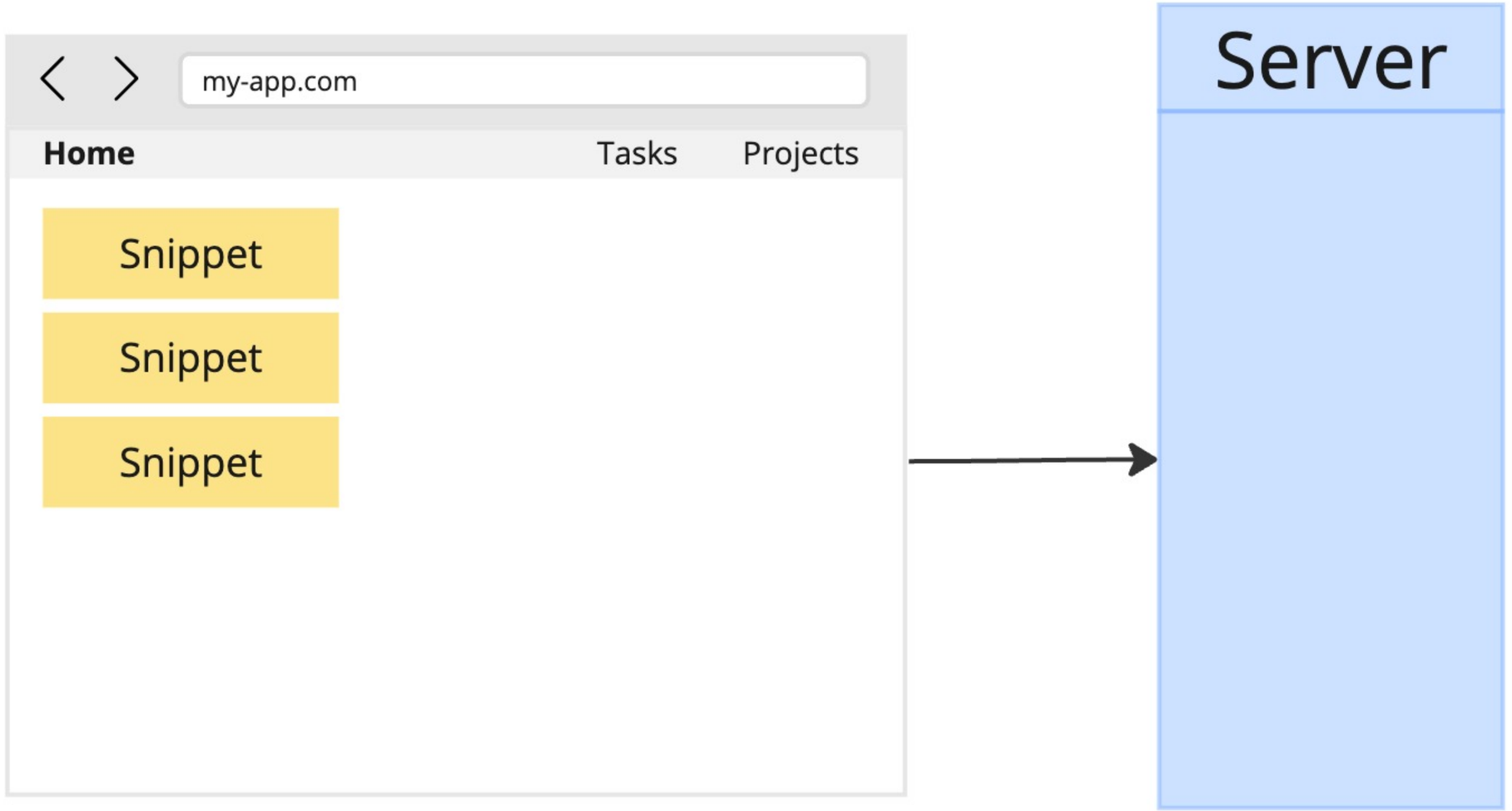

Data fetching seems simple, right?
I wish
Hidden complexity! Seemingly small changes have big impacts!

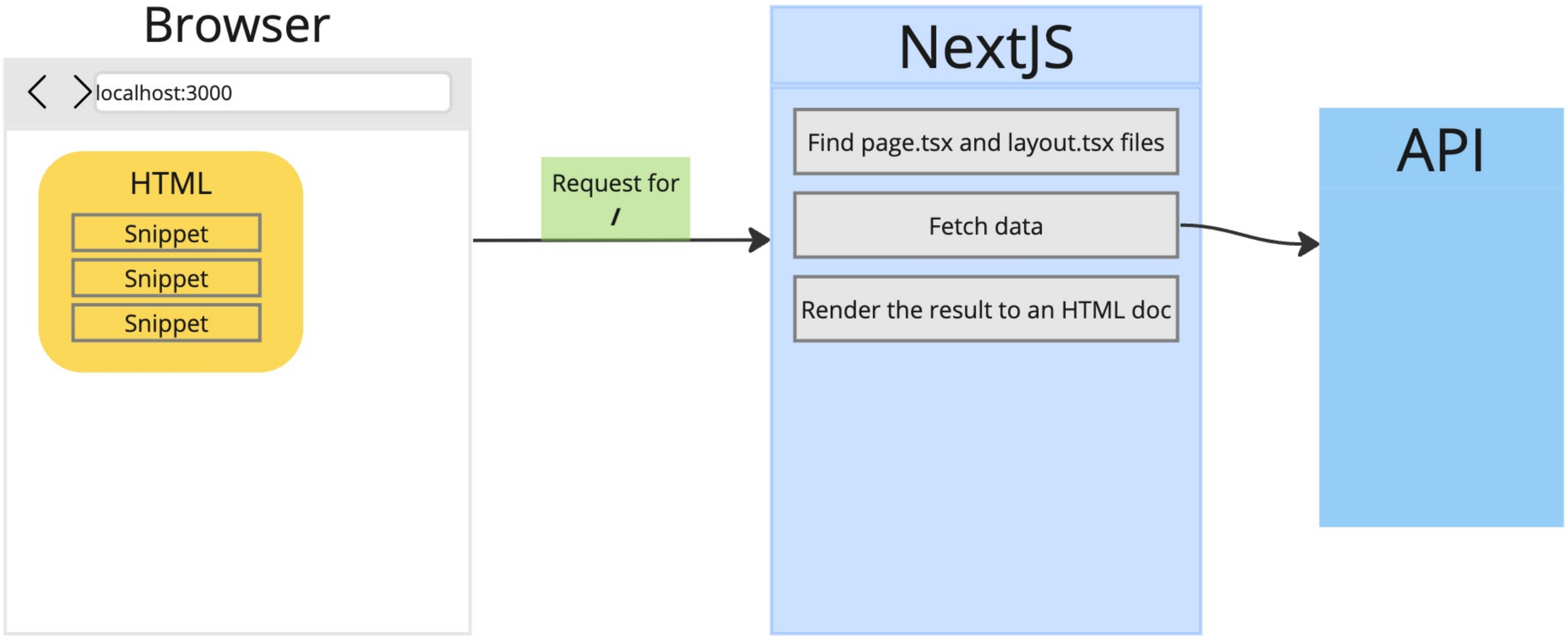
Traditional React Rendering Approach



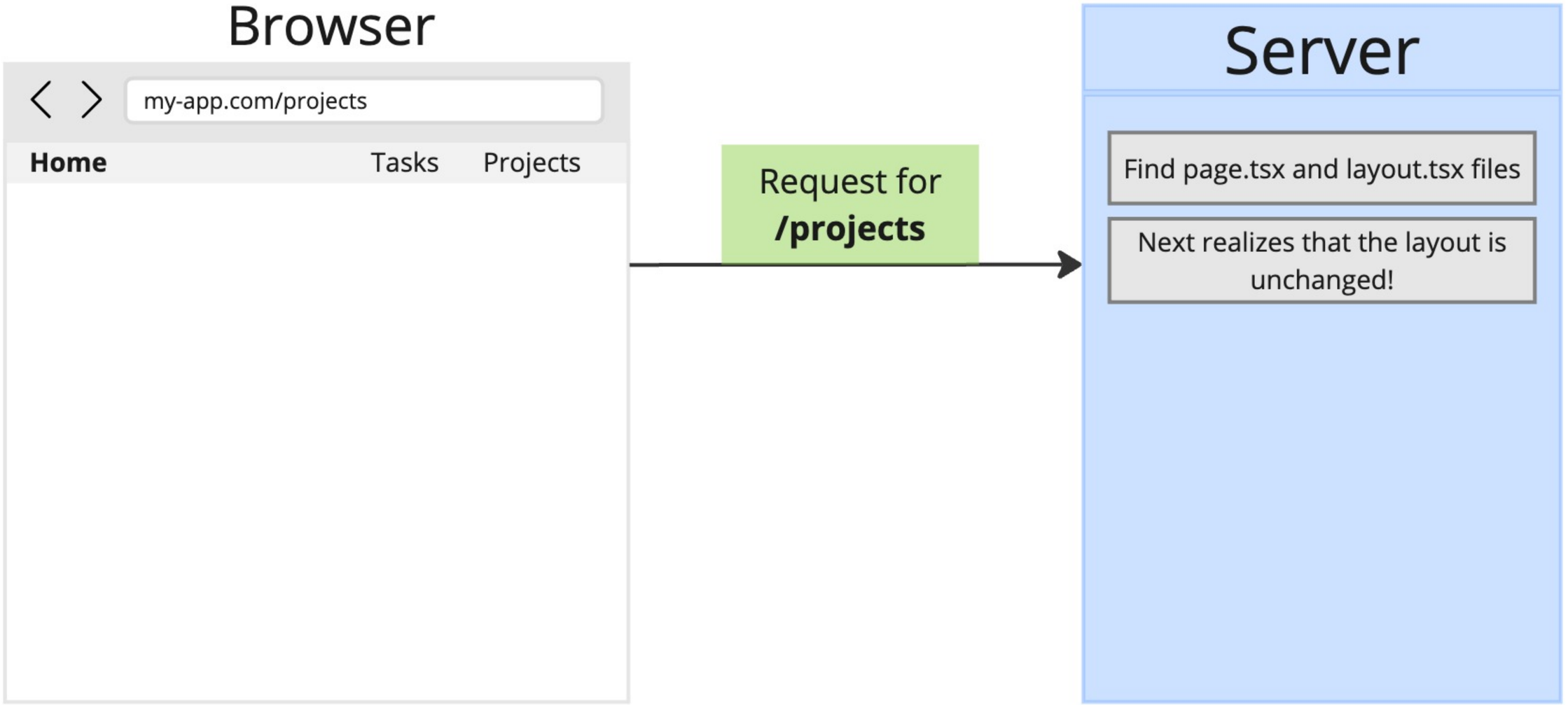
Traditional React Rendering Approach



NextJS Rendering Approach

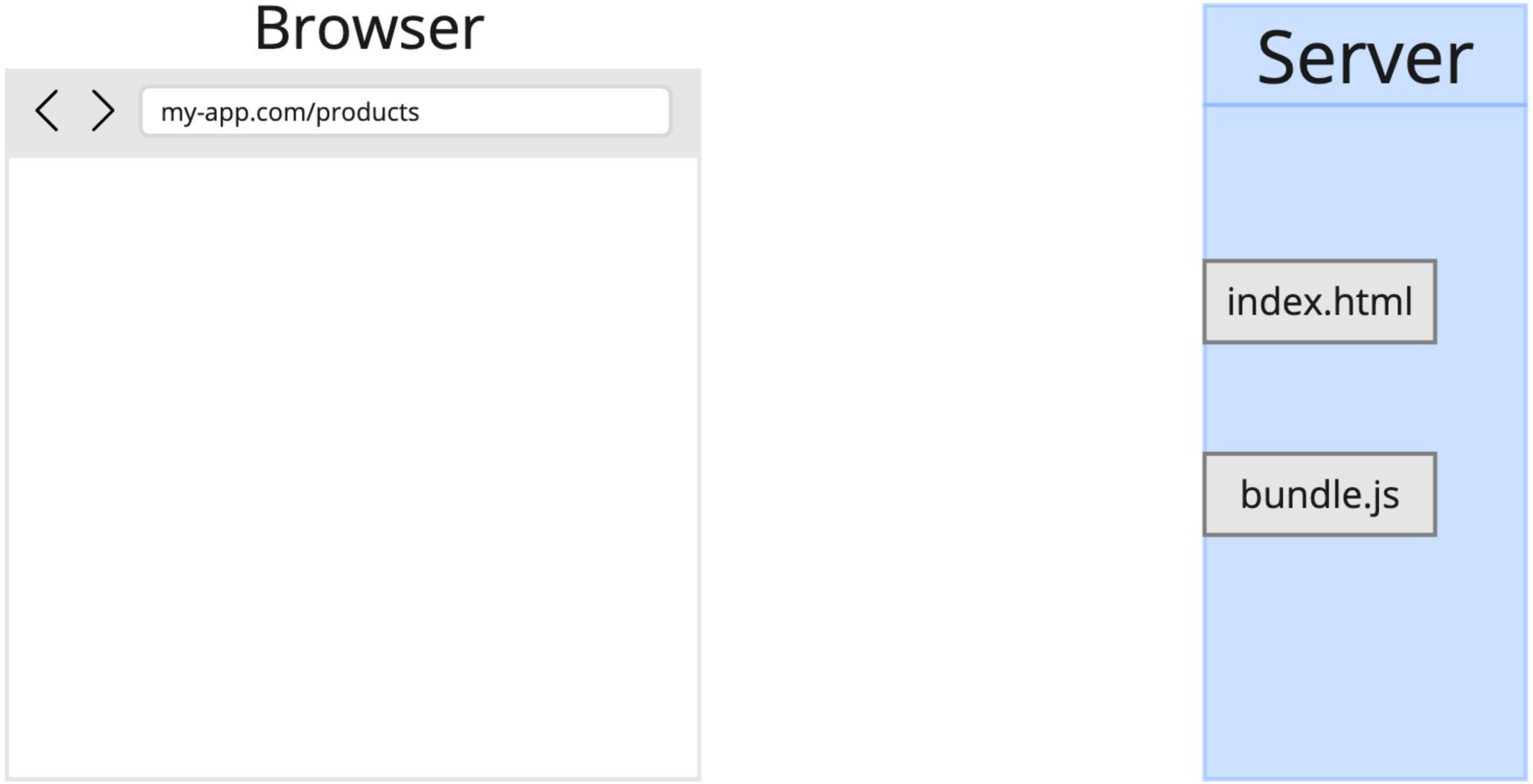


When a user clicks a link



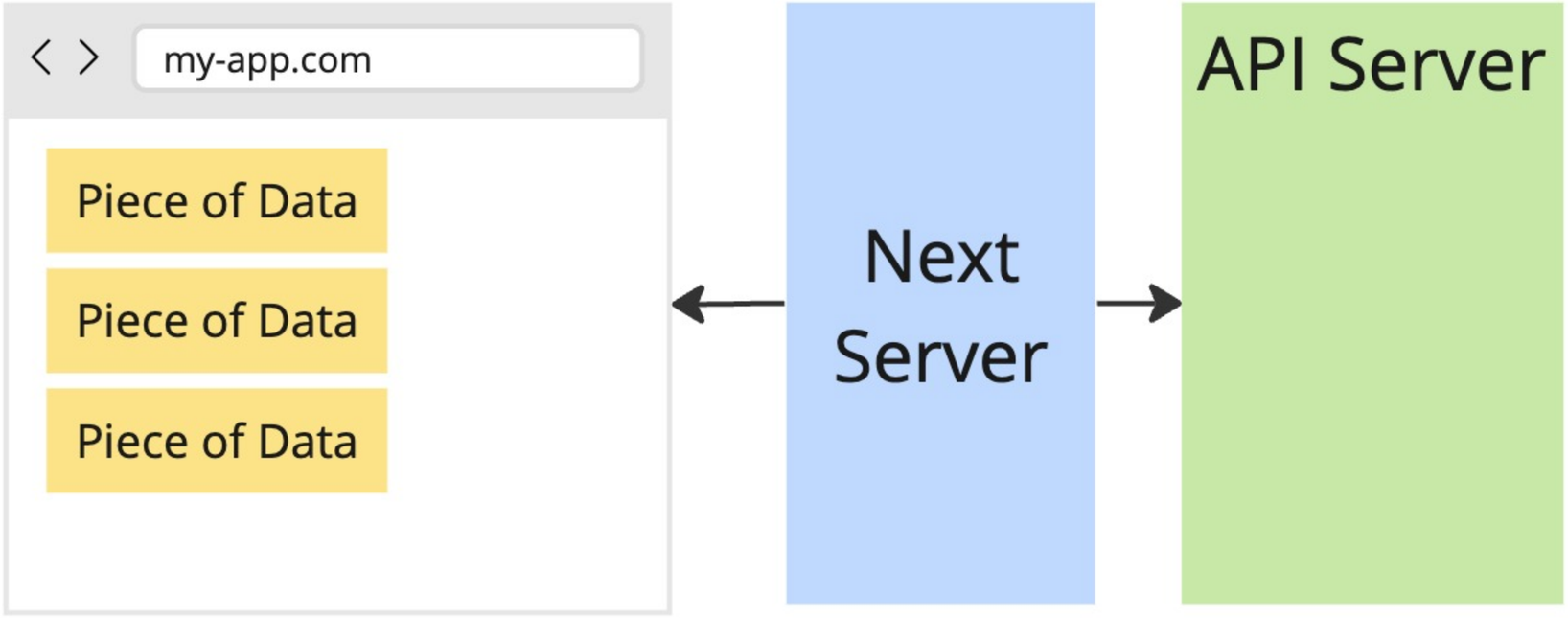
Next JS Rendering Approach

No Data Fetching



First App

Super simple - fetch data from an API and show it on the screen



api-next.fly.dev/snippets

API Server

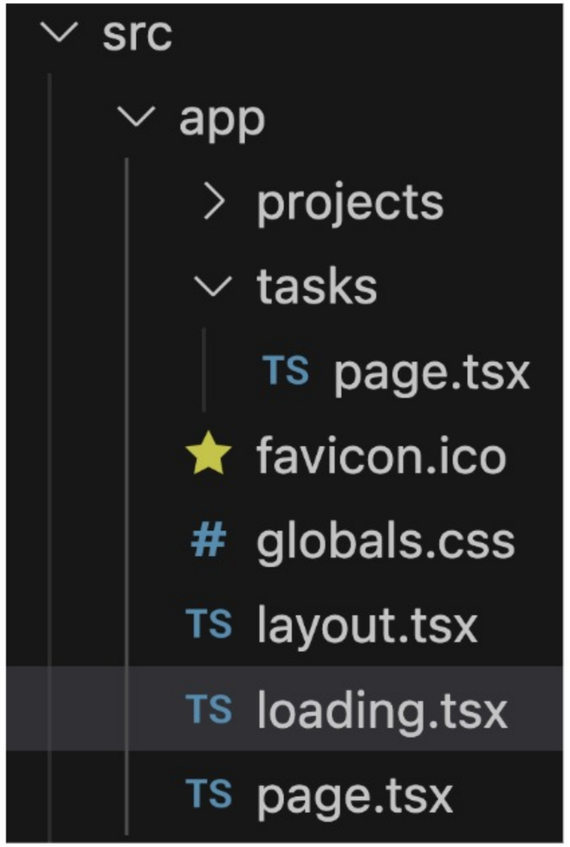
```jsx
import { Suspense } from 'react';

export default function Component() {
    return (
        <Suspense fallback={<div>loading...</div>}
            <ComponentThatLoadsData />
        </Suspense>
    );
}
```

'Suspense' is a component built into React

Watches its child component

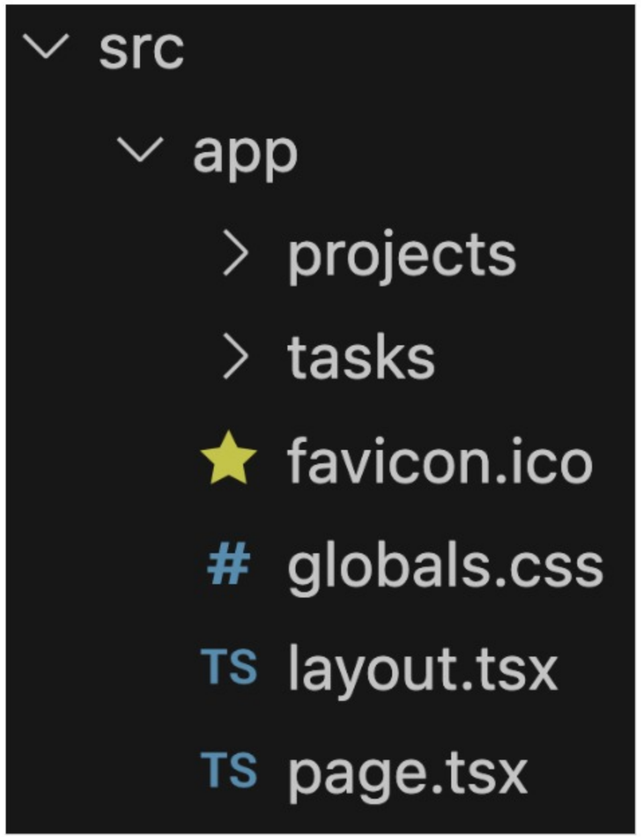
If the child component is being loaded lazily or is data fetching, Suspense shows the component passed to its 'fallback' prop

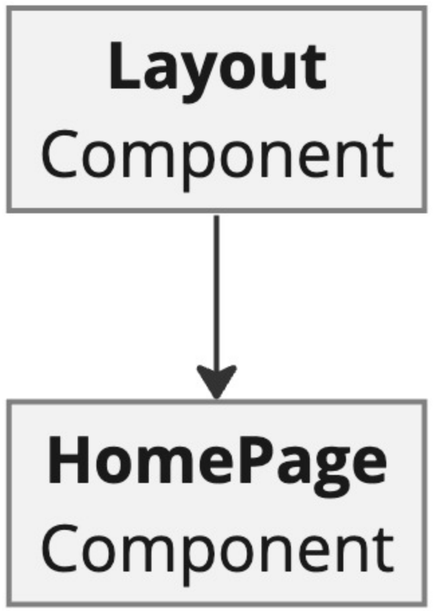


When Next is rendering your page, it finds the closest 'loading.tsx' file

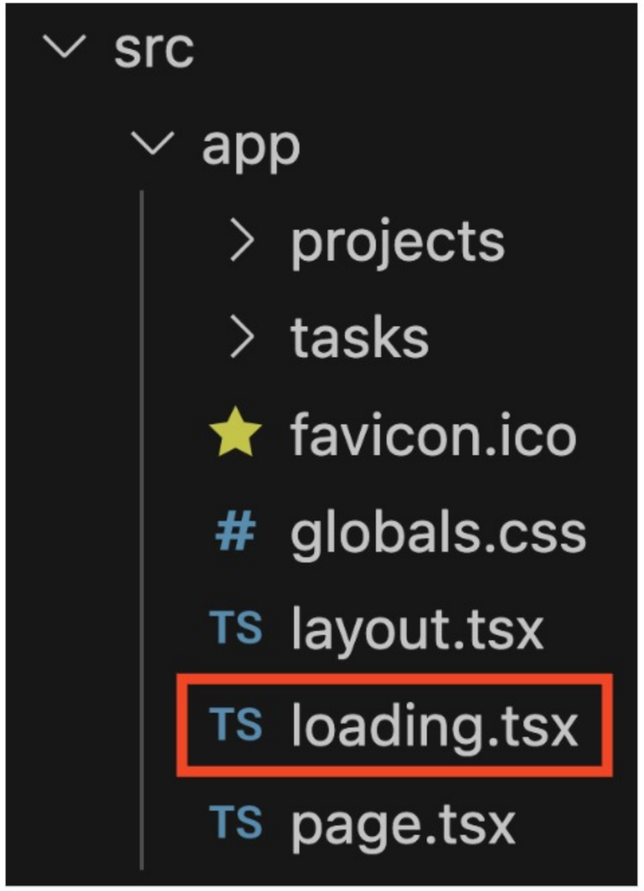
If a 'loading.tsx' is found, then your page is automatically wrapped with a 'Suspense'

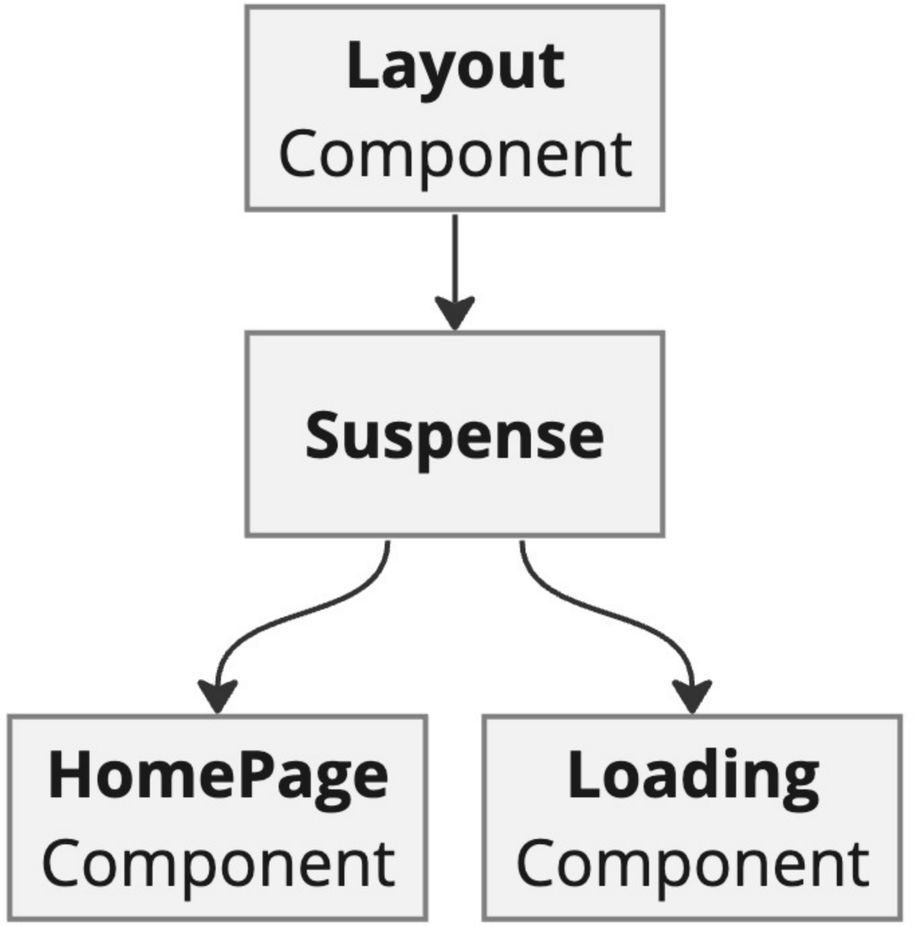
No loading.tsx





With loading.tsx





Exercise!

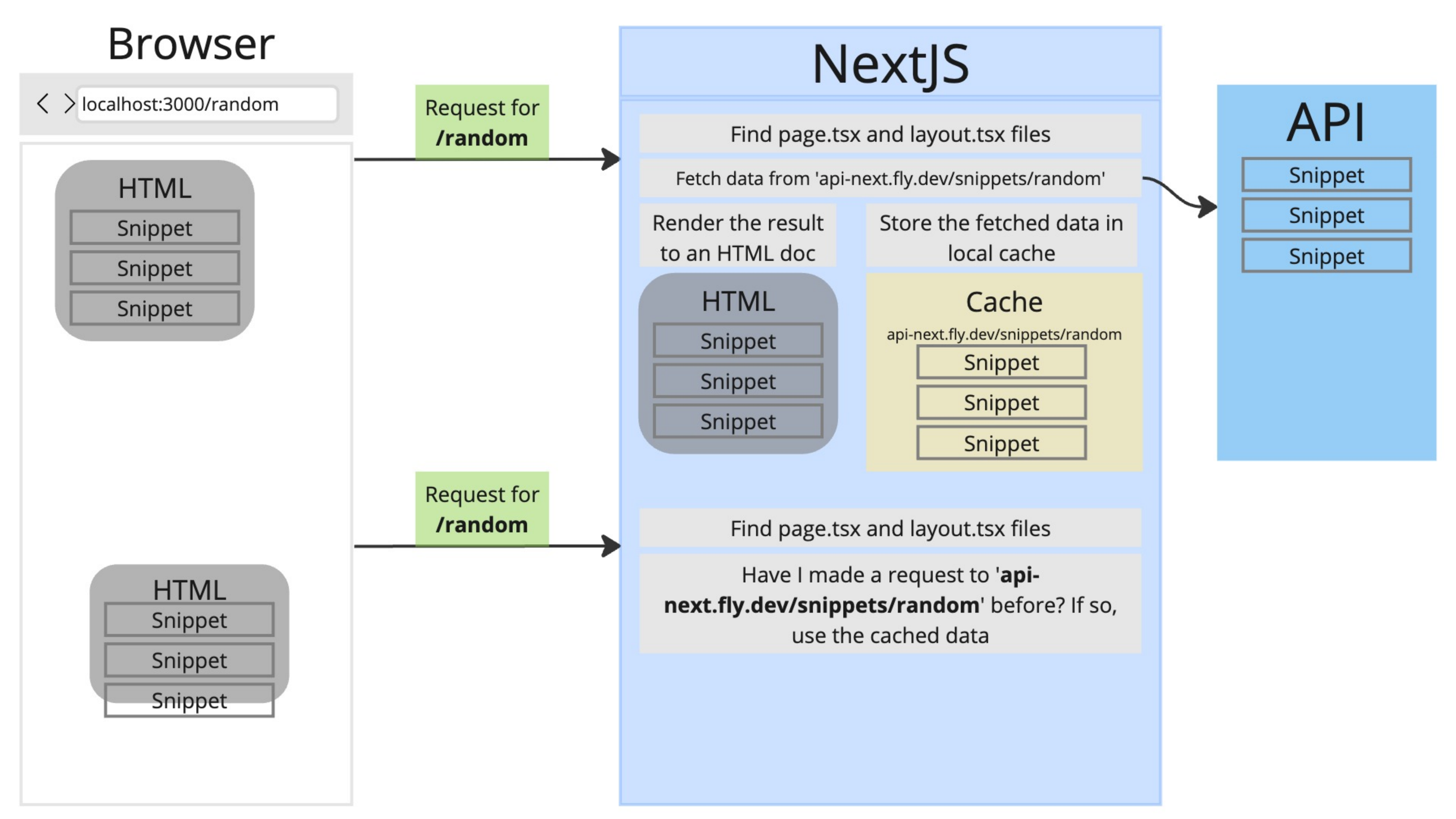
1	Add a new route that a user can access by going to **localhost:3000/random**
2	When a user goes there, fetch a random list of code snippets by making a request to **https://api-next.fly.dev/snippets/random**
3	Render the list of snippets onto the screen, displaying only the "**title**" of each snippet
3.5	Add a link in the Header component to this new page
4	Try to find something *strange* about this new page. Hint - the API gives a **random** list of snippets

Caching

Next implements caching in several locations.

Can lead to unexpected behavior

Data Cache →	Responses from requests made with '**fetch**' are stored and used across requests.
Request Memoization →	Make two or more 'GET' requests with 'fetch' during a user's request to your server? Only one 'GET' is actually executed.
Router Cache →	'Soft' navigation between routes are cached in the browser and reused when a user revisits a page.
Full Route Cache →	At build time, Next decides if your route is ***static*** or ***dynamic***. If it is static, the page is rendered and the result is stored. In production, users are given this pre-rendered result.


Browser
localhost:3000/random
HTML
Snippet
Snippet
Snippet
Request for
/random
NextJS
Find page.tsx and layout.tsx files
Fetch data from 'api-next.fly.dev/snippets/random'
Render the result to an HTML doc
Store the fetched data in local cache
HTML
Snippet
Snippet
Snippet
Cache
api-next.fly.dev/snippets/random
Snippet
Snippet
Snippet
API
Snippet
Snippet
Snippet
Request for
/random
HTML
Snippet
Snippet
Snippet
Find page.tsx and layout.tsx files
Have I made a request to 'api-next.fly.dev/snippets/random' before? If so, use the cached data

Help! My next route is rendering with out-of-date data!

There are several ways to control caching

Time-Based	→	Every 60 seconds, ignore the cached response and fetch new data
On-Demand	→	Forcibly purge a cached response
Disable Caching	→	Don't do any caching at all

Several different ways to change data caching behavior

We are going to briefly touch on each

Super boring and repetitive, but the docs are going to mention them all interchangeably

Time-Based → Every X seconds, ignore the cached response and fetch new data

Option #1

Refresh for a *single fetch statement*

```
export default async function Page() {
    // Dumps cached data every 3s
    await fetch('/snippets/random', {
        next: { revalidate: 3 }
    })

    // Always uses cached data
    await fetch('/tasks/random')
}
```

Option #2

Refresh for a *all fetches in a route*

```
export const revalidate = 3;

export default async function Page() {
    // Dumps cached data every 3s
    await fetch('/snippets/random')

    // Dumps cached data every 3s
    await fetch('/tasks/random')
}
```

On-Demand → Forcibly purge a cached response

Option #1

Dump cache for everything in a page

```
import { revalidatePath } from "next/cache";

// When we think data that the '/random'
// route uses has changed...
revalidatePath('/random');
```

Option #2

Dump cache for all data with a certain tag

```
import { revalidateTag } from "next/cache";

// When we think data with tag 'snippets'
// has changed...
revalidateTag('snippets');


export default async function Page() {
    await fetch('/snippets/random', {
        next: { tags: ['snippets'] }
    })
}
```

Disable Caching

→ Don't do any caching at all

Option #1

Disable all caching for a route

```
export const revalidate = 0;

export default async function Page() {
    // Never caches!
    await fetch('/snippets/random')

    // Never caches!
    await fetch('/tasks/random')
}
```

Option #2

Disable all caching for a route

```
export const dynamic = "force-dynamic";

export default async function Page() {
    // Never caches!
    await fetch('/snippets/random')

    // Never caches!
    await fetch('/tasks/random')
}
```

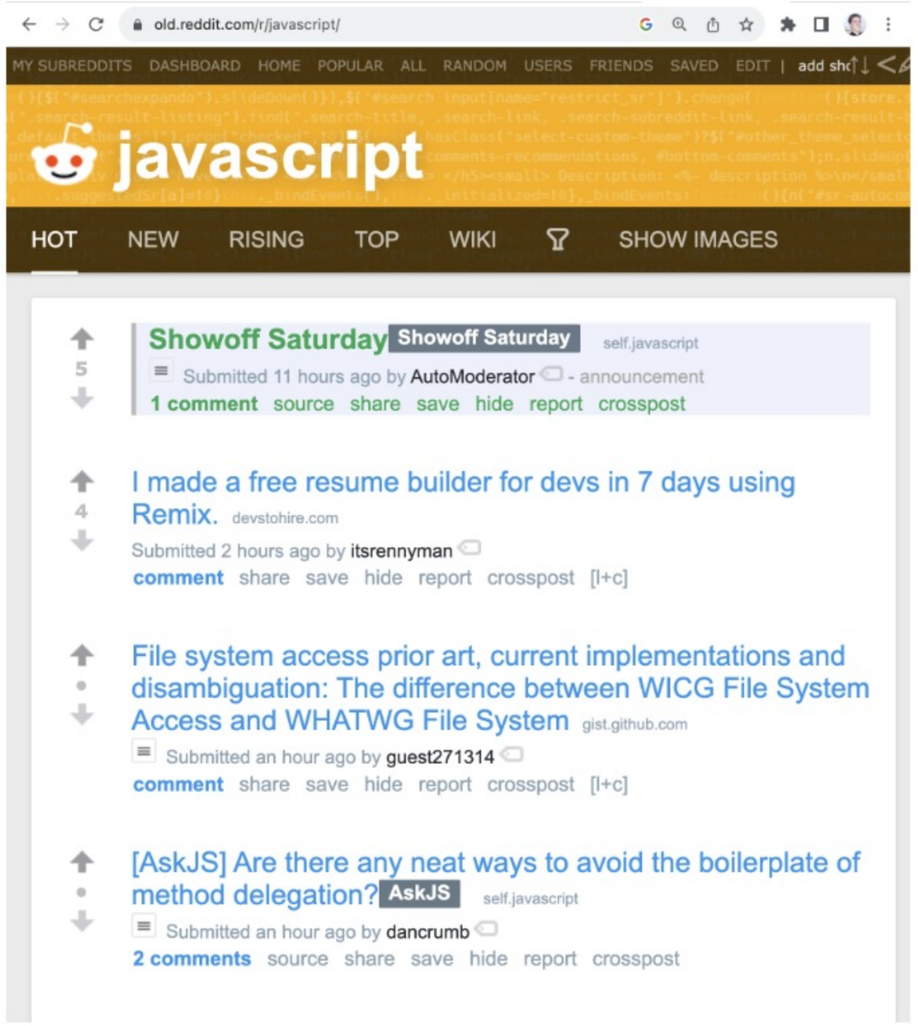
Option #3

Disable all caching for a particular fetch

```
export default async function Page() {
    // Never caches!
    await fetch('/snippets/random', {
        cache: 'no-store'
    })

    // Caches as usual
    await fetch('/tasks/random')
}
```

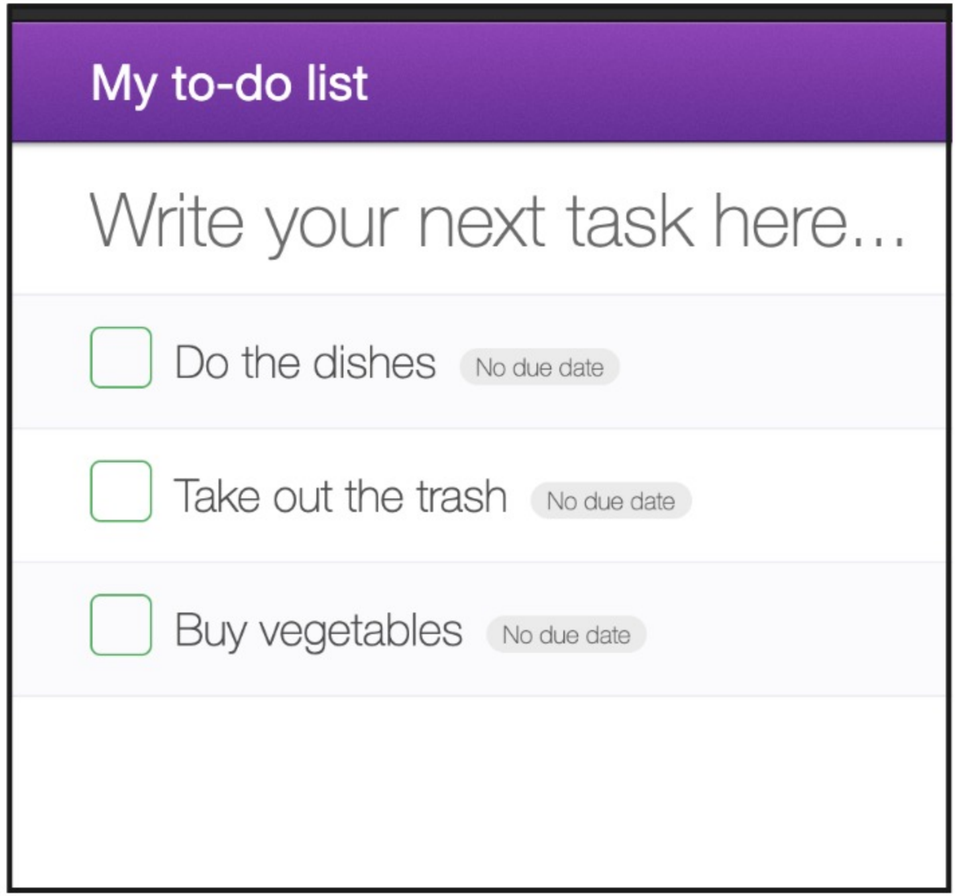
Time-Based → Every X seconds, ignore the cached response and fetch new data



Front page of a social media site

Data is changing all the time - only get the top posts every 10-30 seconds

On-Demand → Forcibly purge a cached response



Any app where we know **when** data changes **and** the user expects to see up-to-date data

Disable Caching → Don't do any caching at all

NextJS → Outside API

Any app where we
(don't know when data changes
or
when we expect the data to be different with every request)
and
the user still expects to see up to date data

Home Page

↓

Fetches a list of code snippets that **never changes**

↓

Default behavior is to cache everything. Do we need to change that? If so, to what strategy?

Random Page

↓

Fetches a list of code snippets that **changes with every request**

↓

Default behavior is to cache everything. Do we need to change that? If so, to what strategy?

Hard Navigation

Browser makes a full request to the Nextjs server asking for an HTML document

Occurs when...

user types a route directly into address bar

or

user opens a <Link /> into a new tab

or

user initially navigates to our site

or

we write code to specifically do hard navigation

Soft Navigation

Next code running in the browser makes a request asking for resources needed to show a new page

Occurs when...

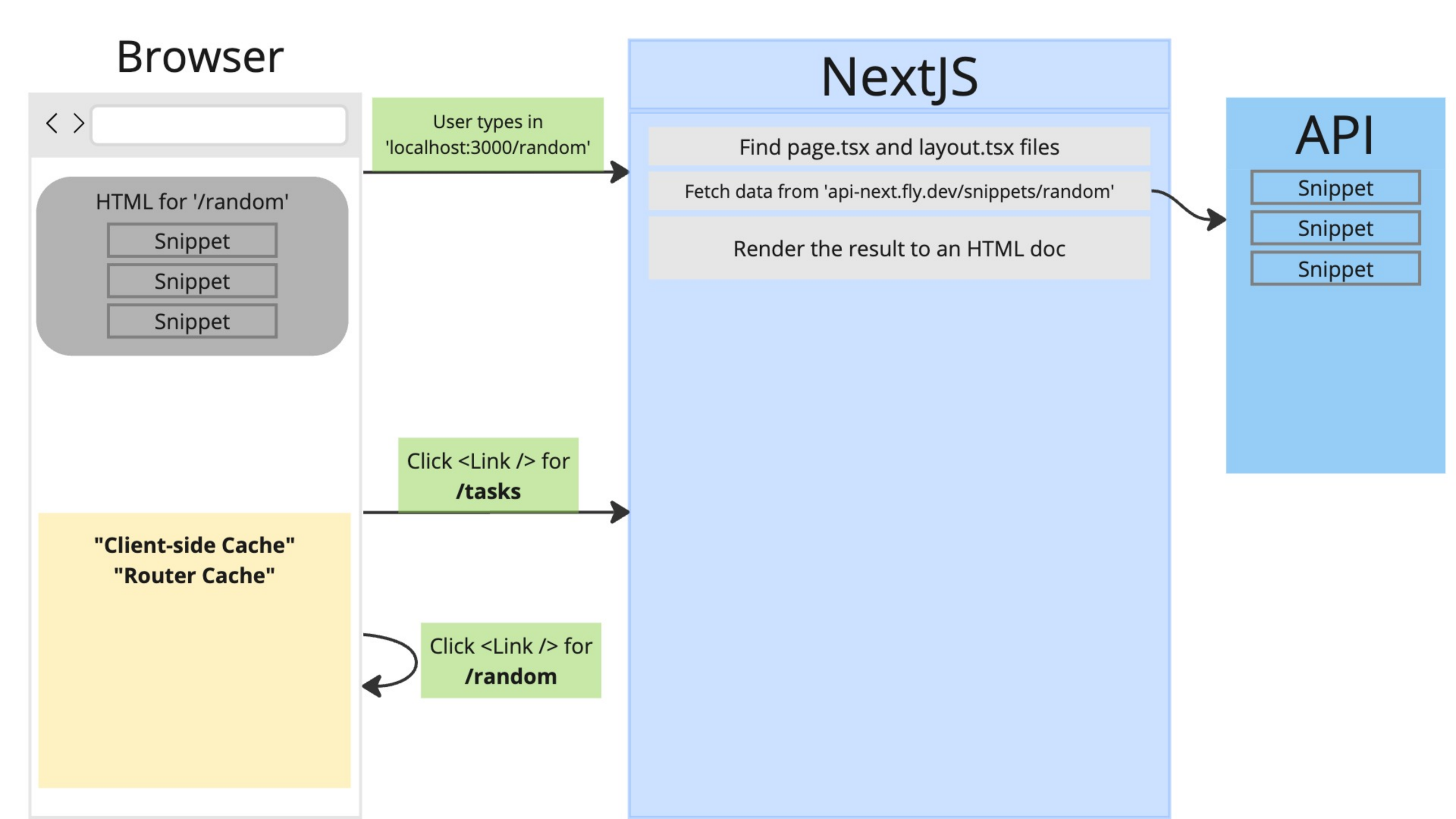
user clicks on a <Link />

or

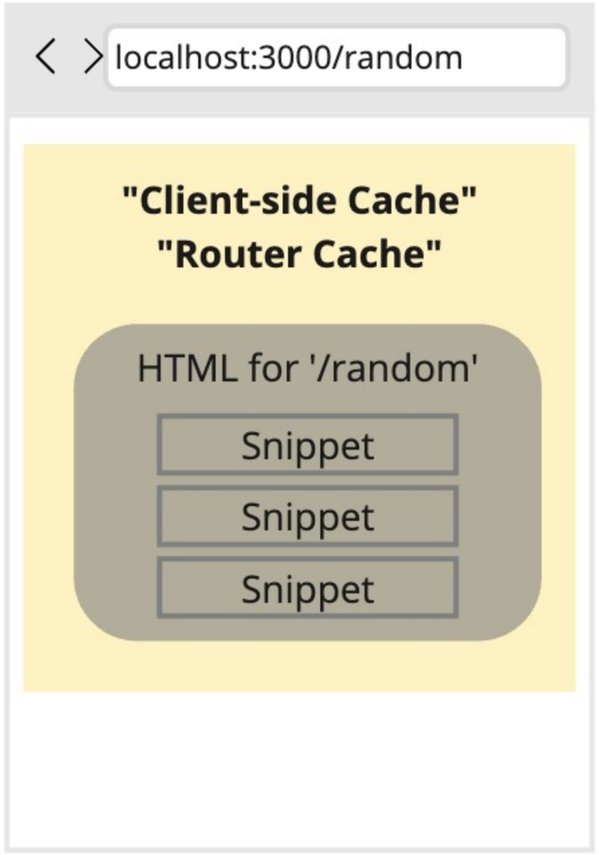
we write code to navigate the user around

or

user clicks forward/back buttons in browser


Browser
HTML for '/random'
Snippet
Snippet
Snippet
"Client-side Cache"
"Router Cache"
User types in 'localhost:3000/random'
Click <Link /> for /tasks
Click <Link /> for /random
NextJS
Find page.tsx and layout.tsx files
Fetch data from 'api-next.fly.dev/snippets/random'
Render the result to an HTML doc
API
Snippet
Snippet
Snippet

Managing This Cache



Timer

Router cache automatically clears records older than 30 seconds

At the time I'm recording this, you can't modify that time

You will soon be able to add a config variable of 'staletime' to your page.tsx file to adjust it

On Demand

Use a 'server action' along with 'revalidatePath' or 'revalidateTag'

More on server actions later

Use 'router.refresh()'

More on router.refresh() later