Introducing Server Components

Peter Kellner

Developer and Author

peterkellner.net | linkedin.com/in/peterkellner99 Mastodon: @pkellner@peterkellner.net

Server Components

Concurrent rendering engine released with React 18

Streaming, improved with concurrent rendering

Server Components

Concurrent rendering engine released with React 18

Streaming, improved with concurrent rendering

Server Components

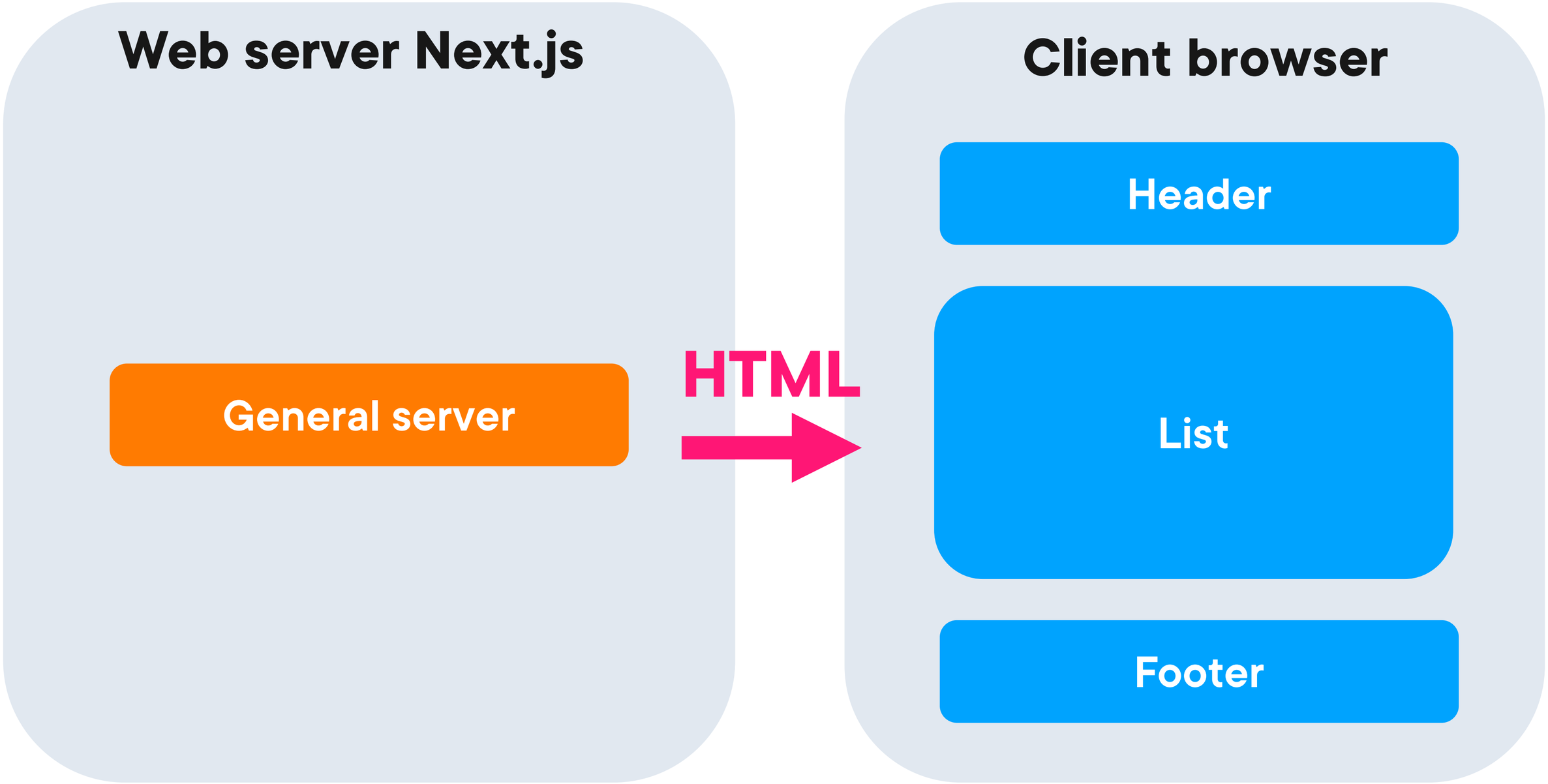
Concurrent rendering engine released with React 18

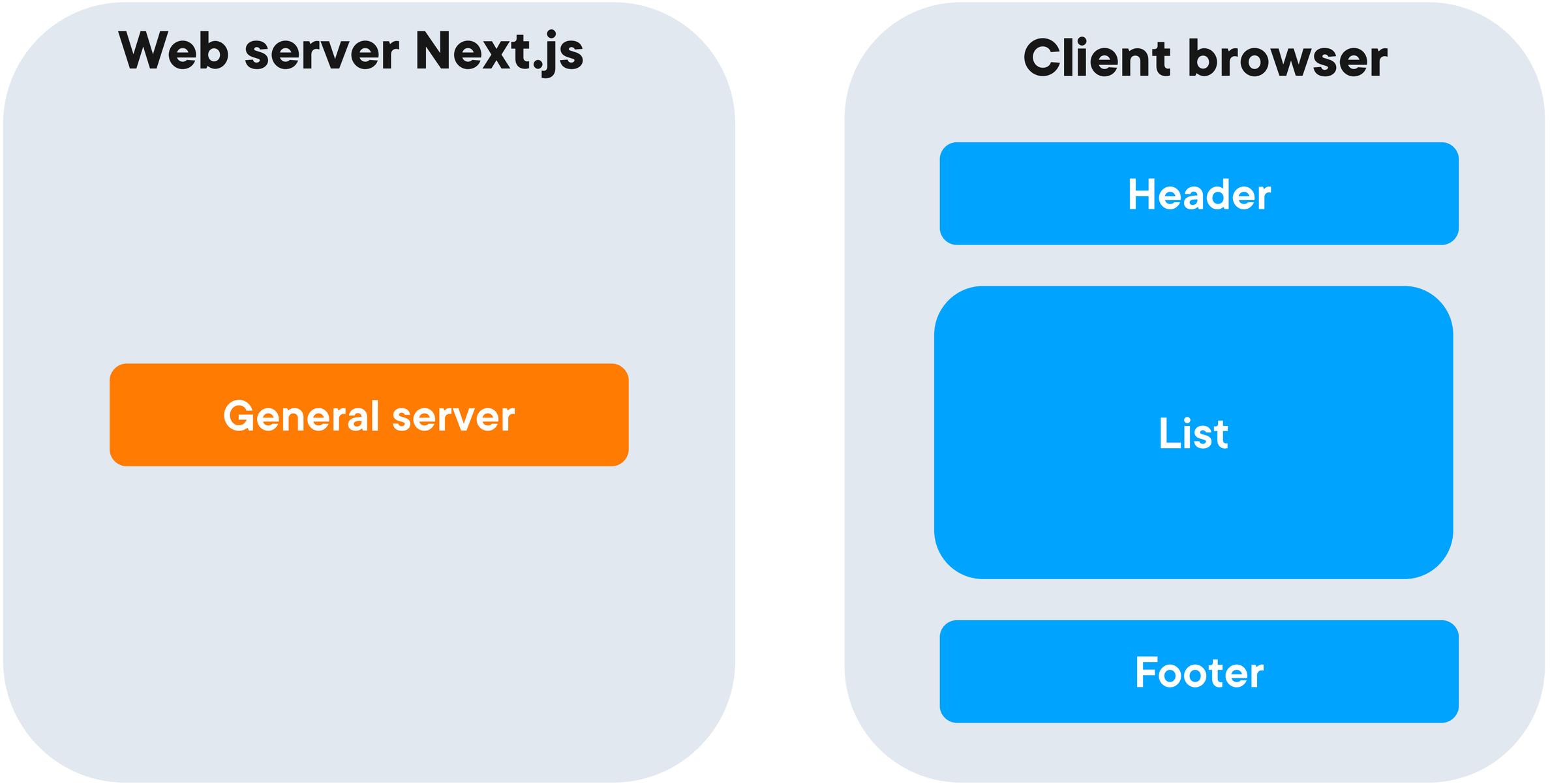
Streaming, improved with concurrent rendering

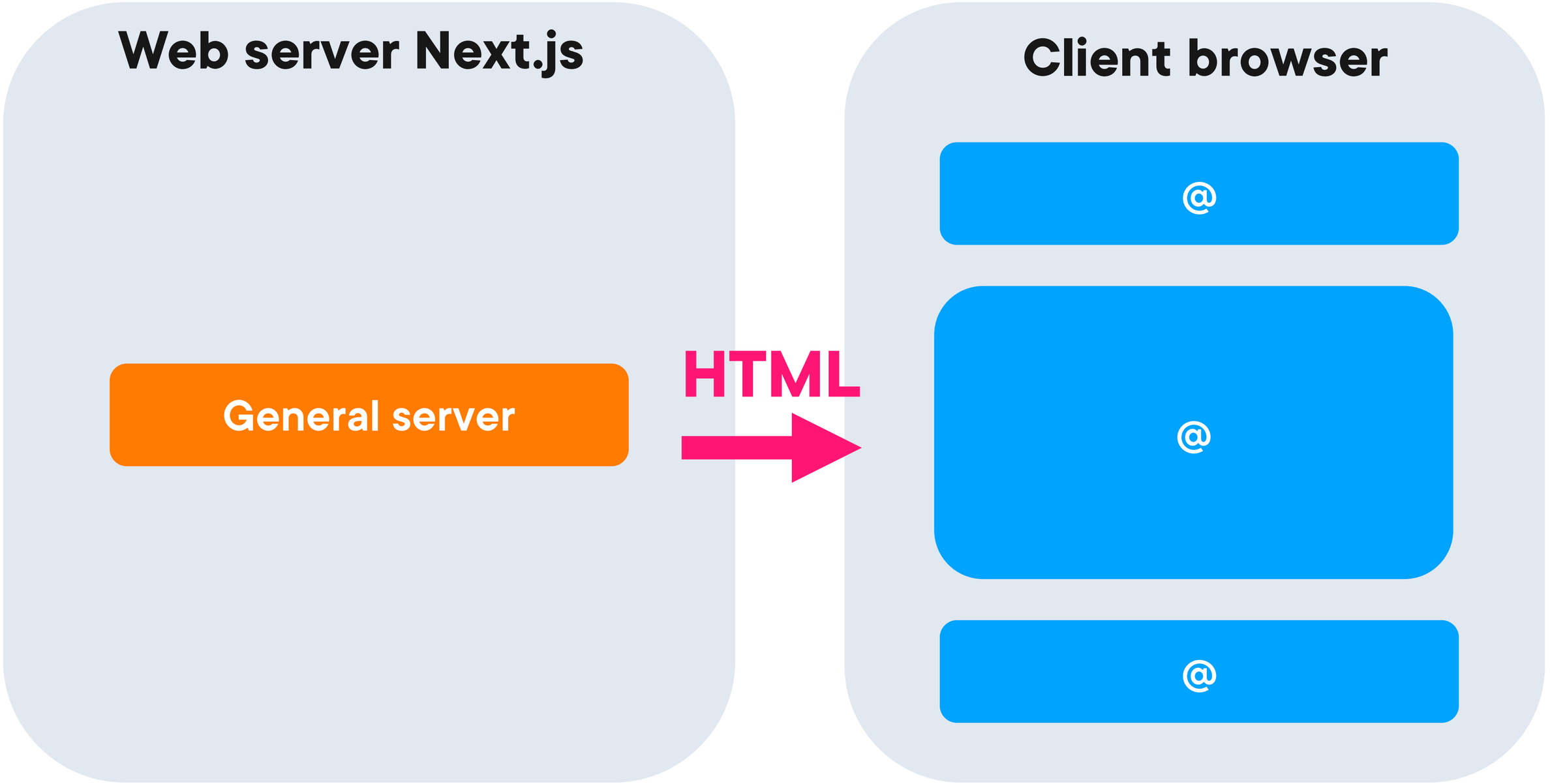
Server-side Rendering (SSR)

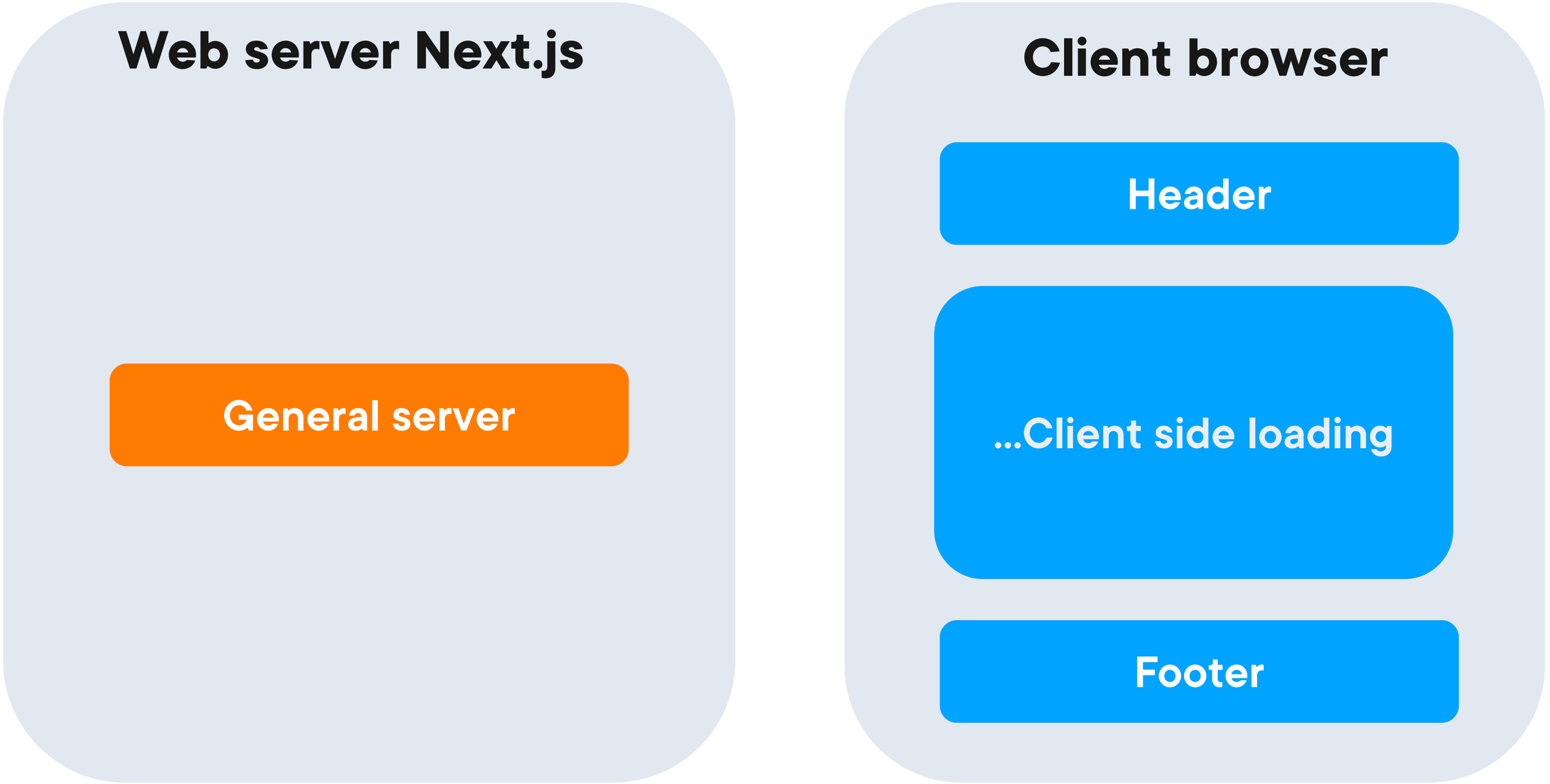
1. Browser does a GET request
2. Server starts Node
3. ReactDOM called in Node to render HTML
4. Text block is returned to browser
5. JavaScript hydrates browser HTML

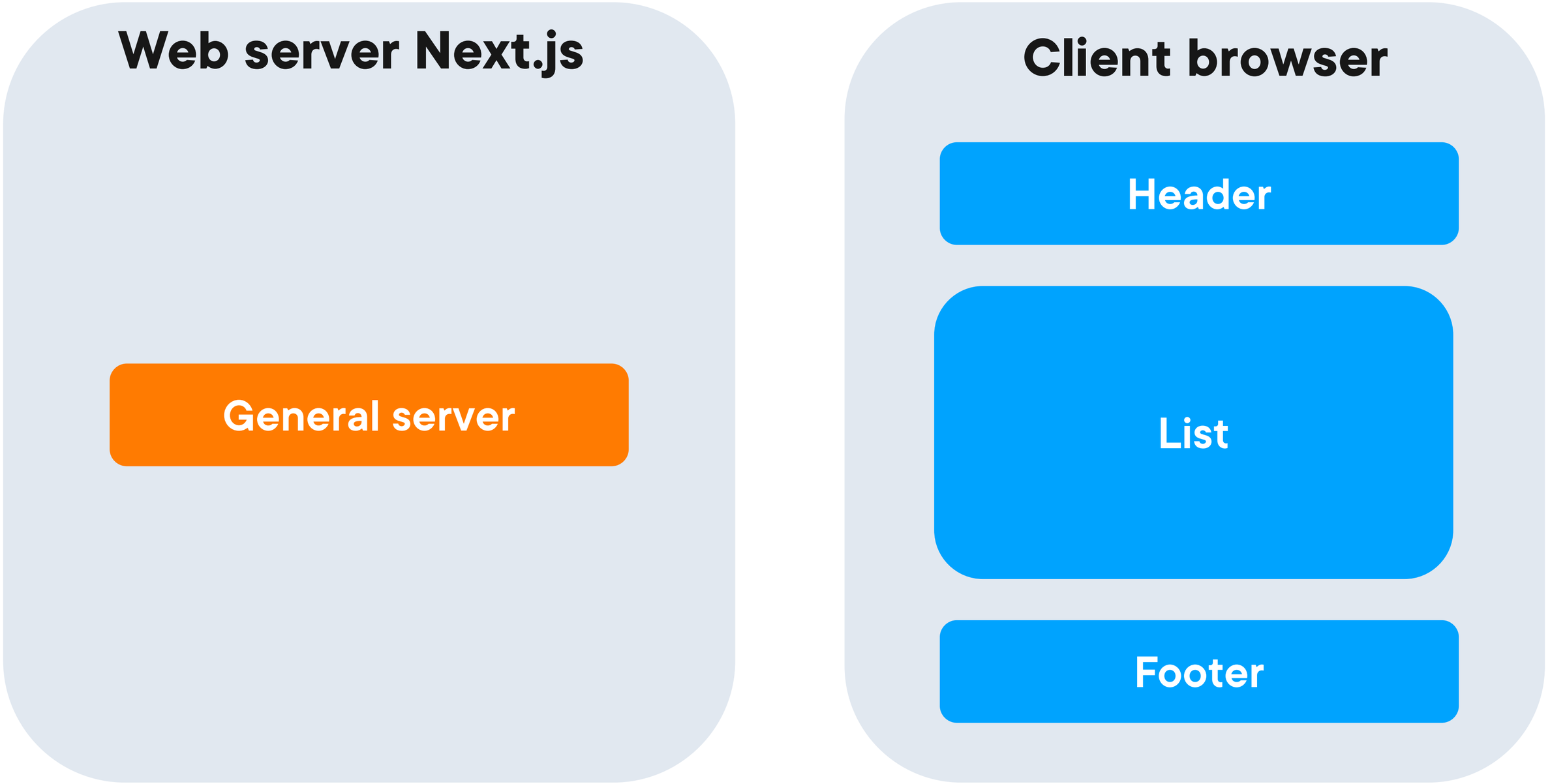
Server-side Rendering is like a single Server Component app

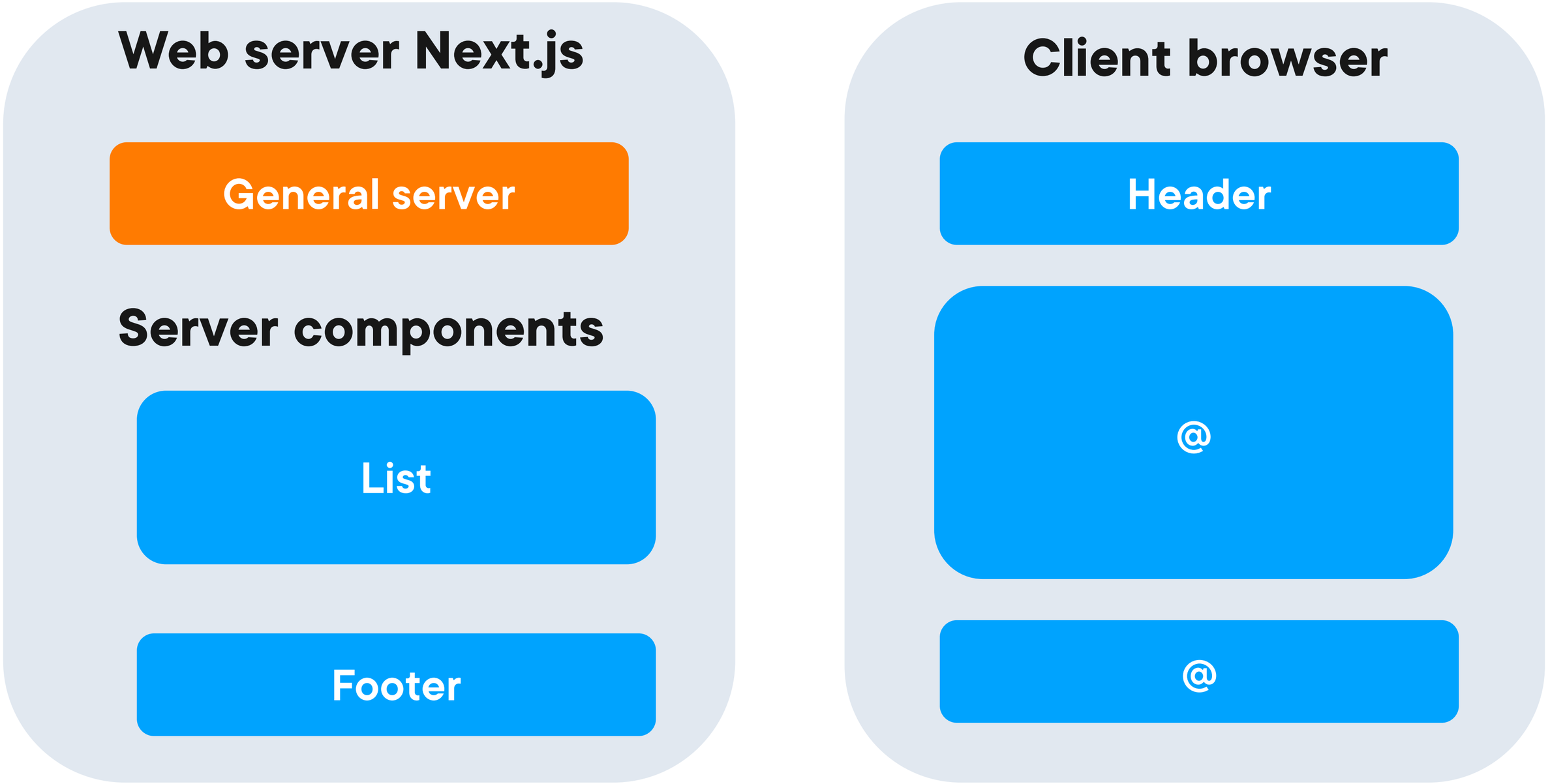

Web server Next.js
General server
HTML
Client browser
Header
List
Footer

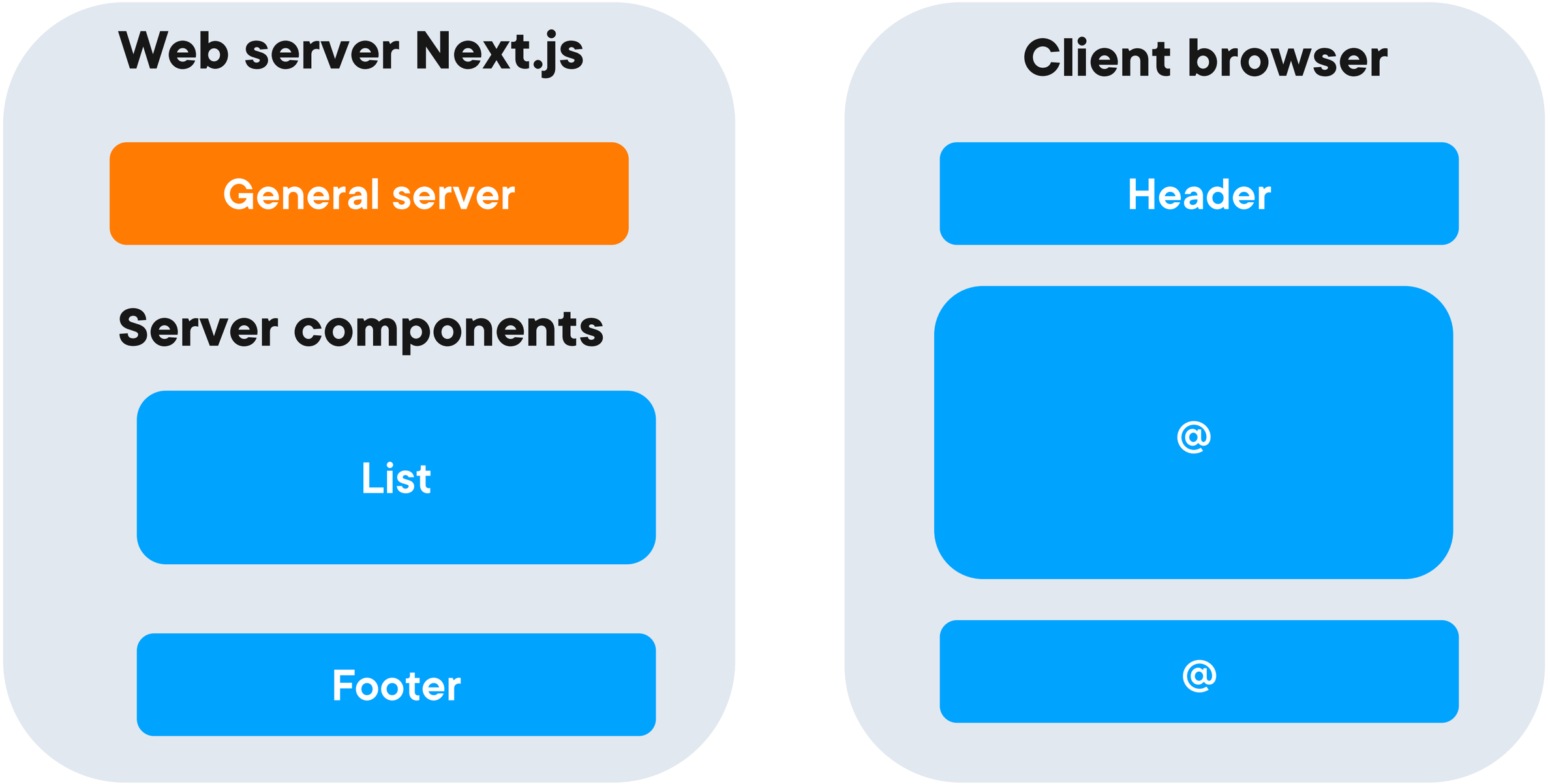

Web server Next.js
General server
Client browser
Header
List
Footer

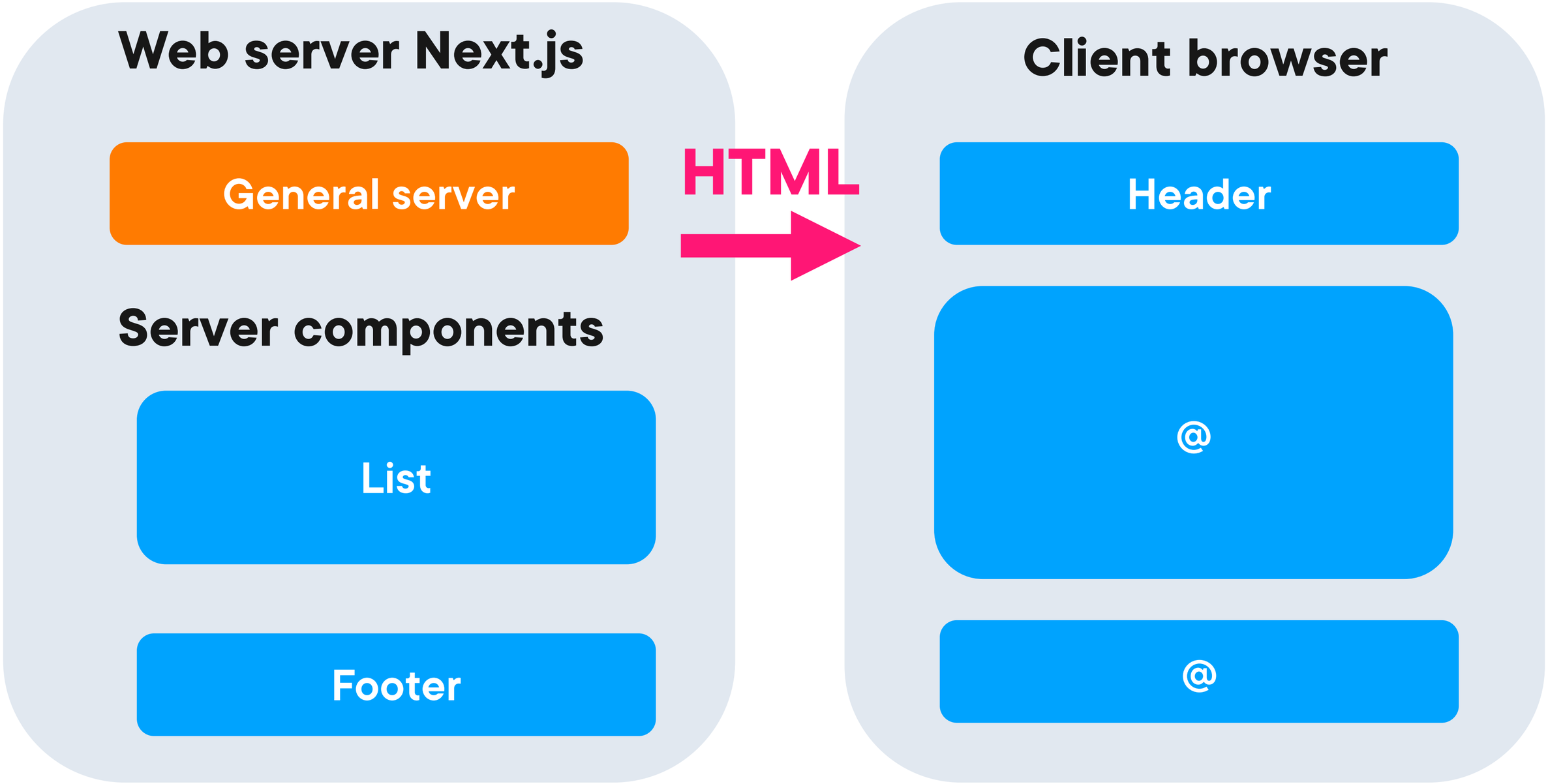

Web server Next.js
General server
HTML
Client browser
@
@
@

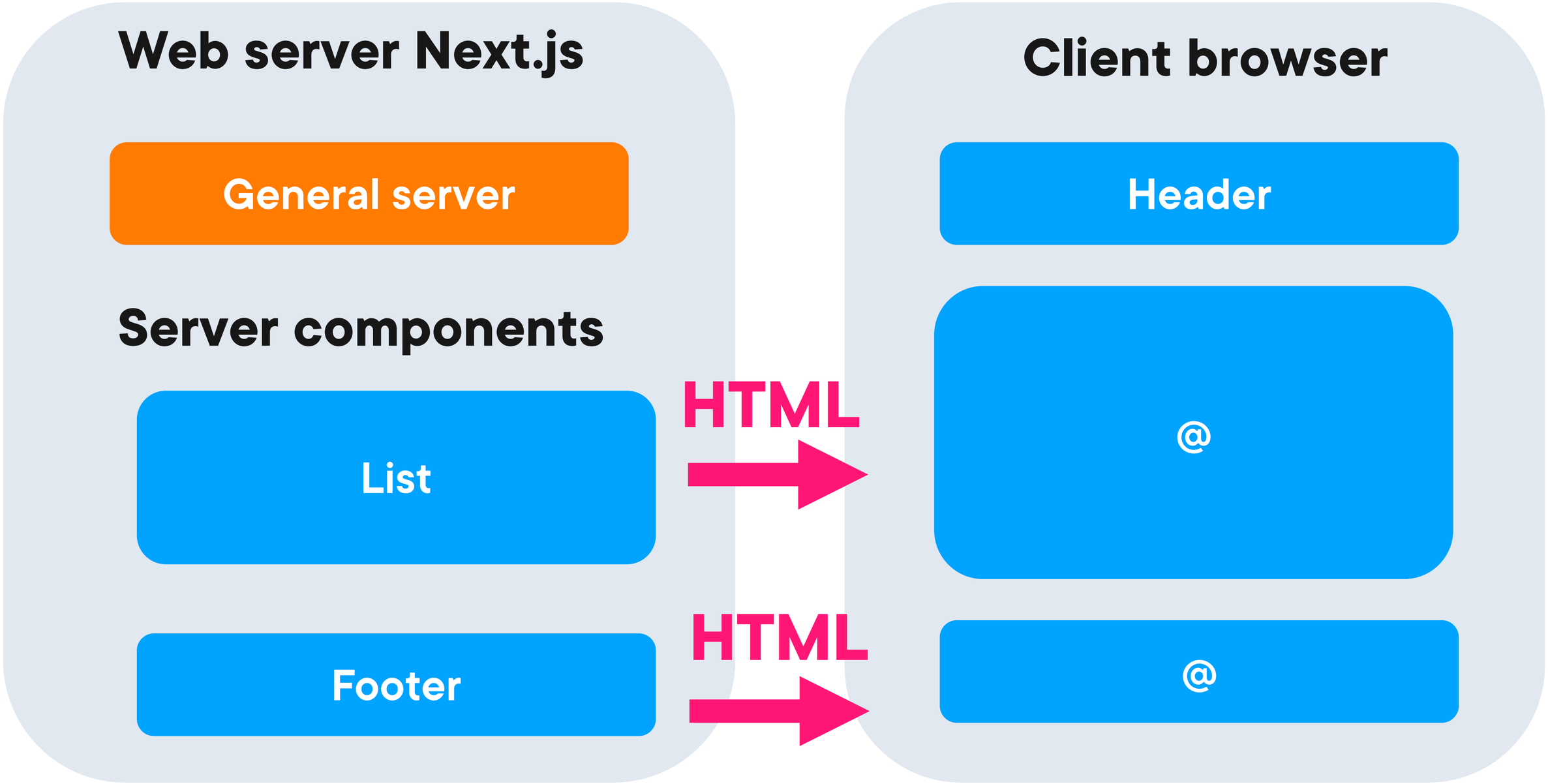

Web server Next.js
General server
Client browser
Header
...Client side loading
Footer

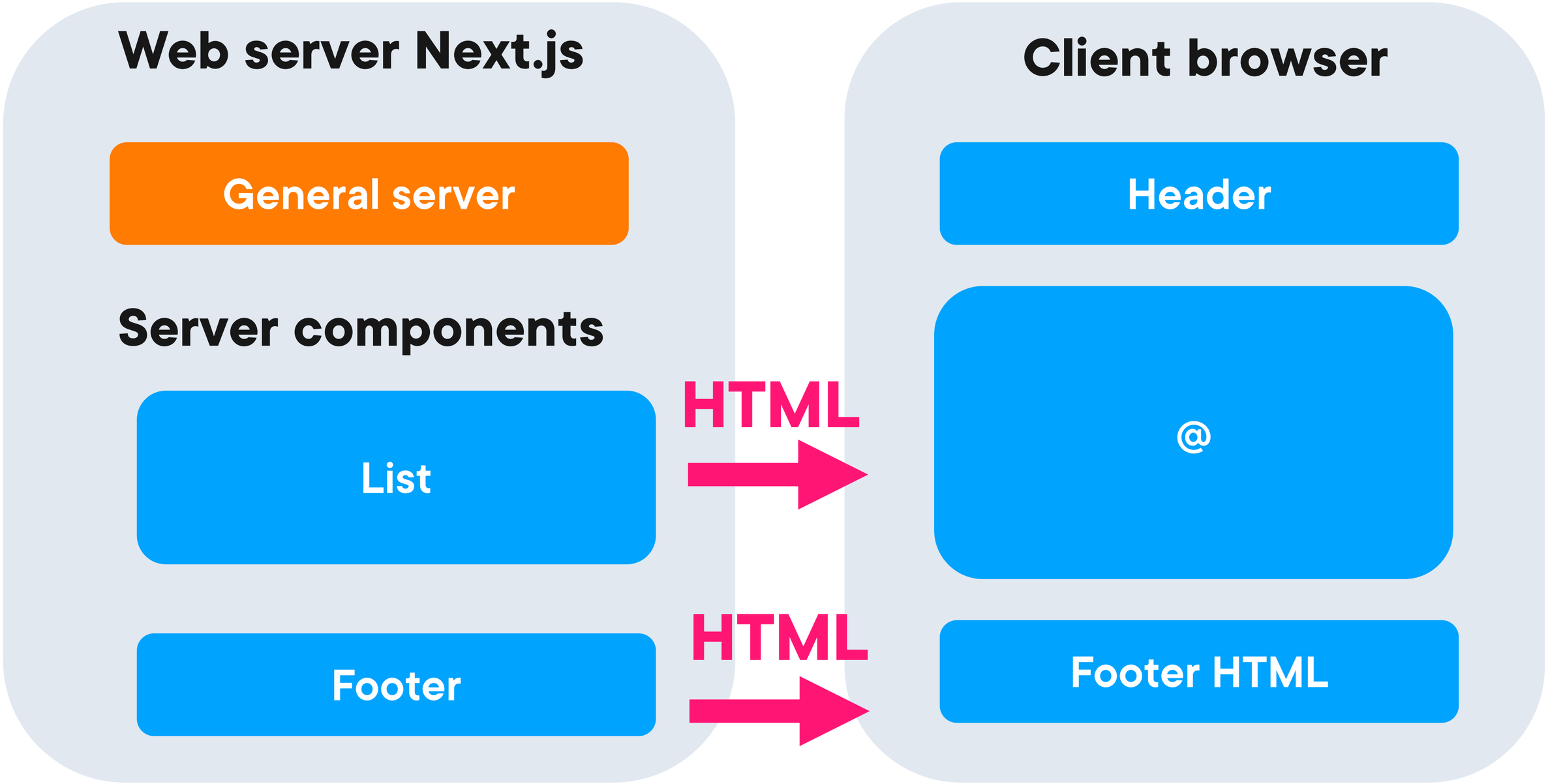

Web server Next.js
General server
Client browser
Header
List
Footer

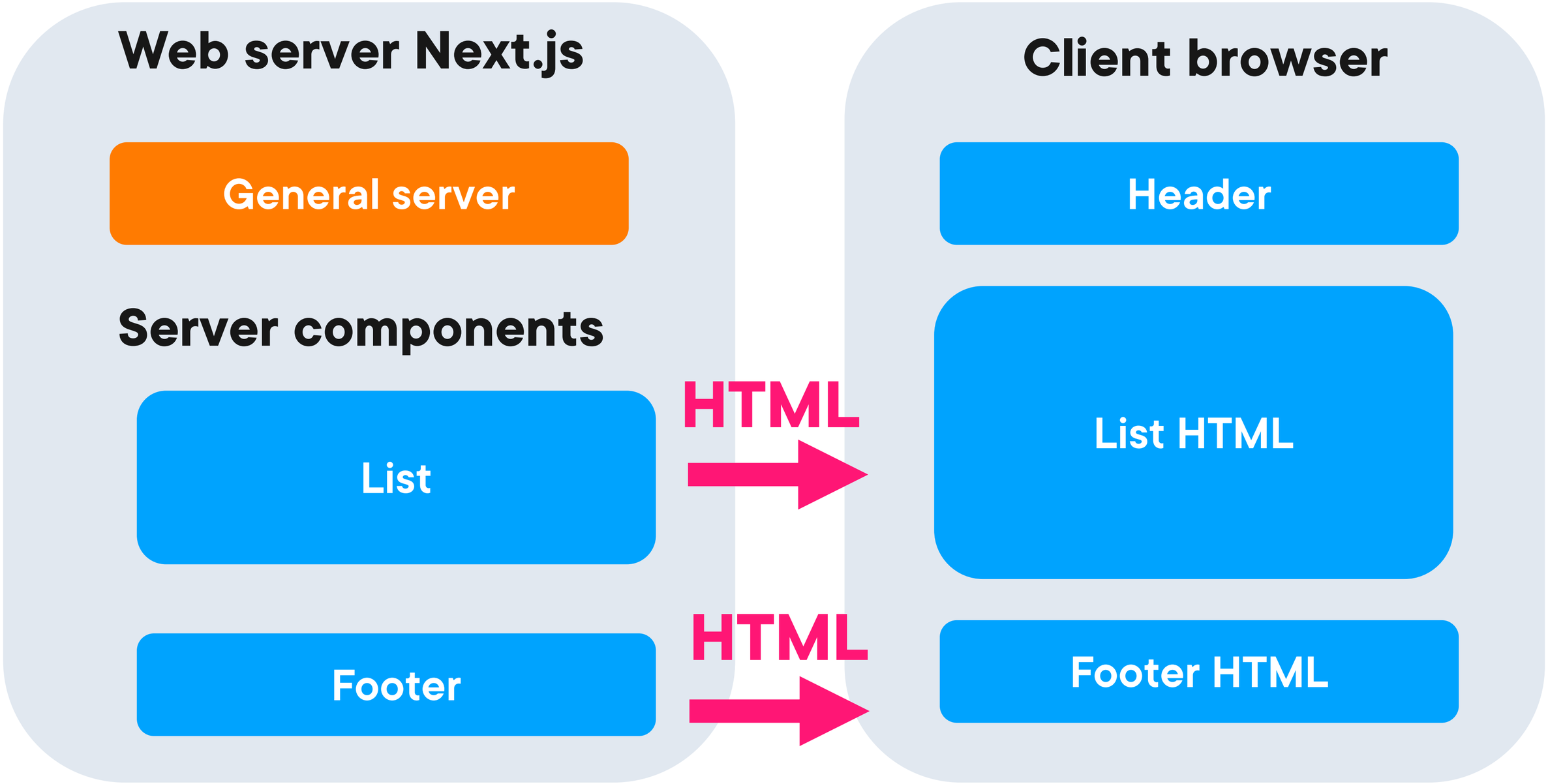

Web server Next.js
General server
Server components
List
Footer
Client browser
Header
@
@

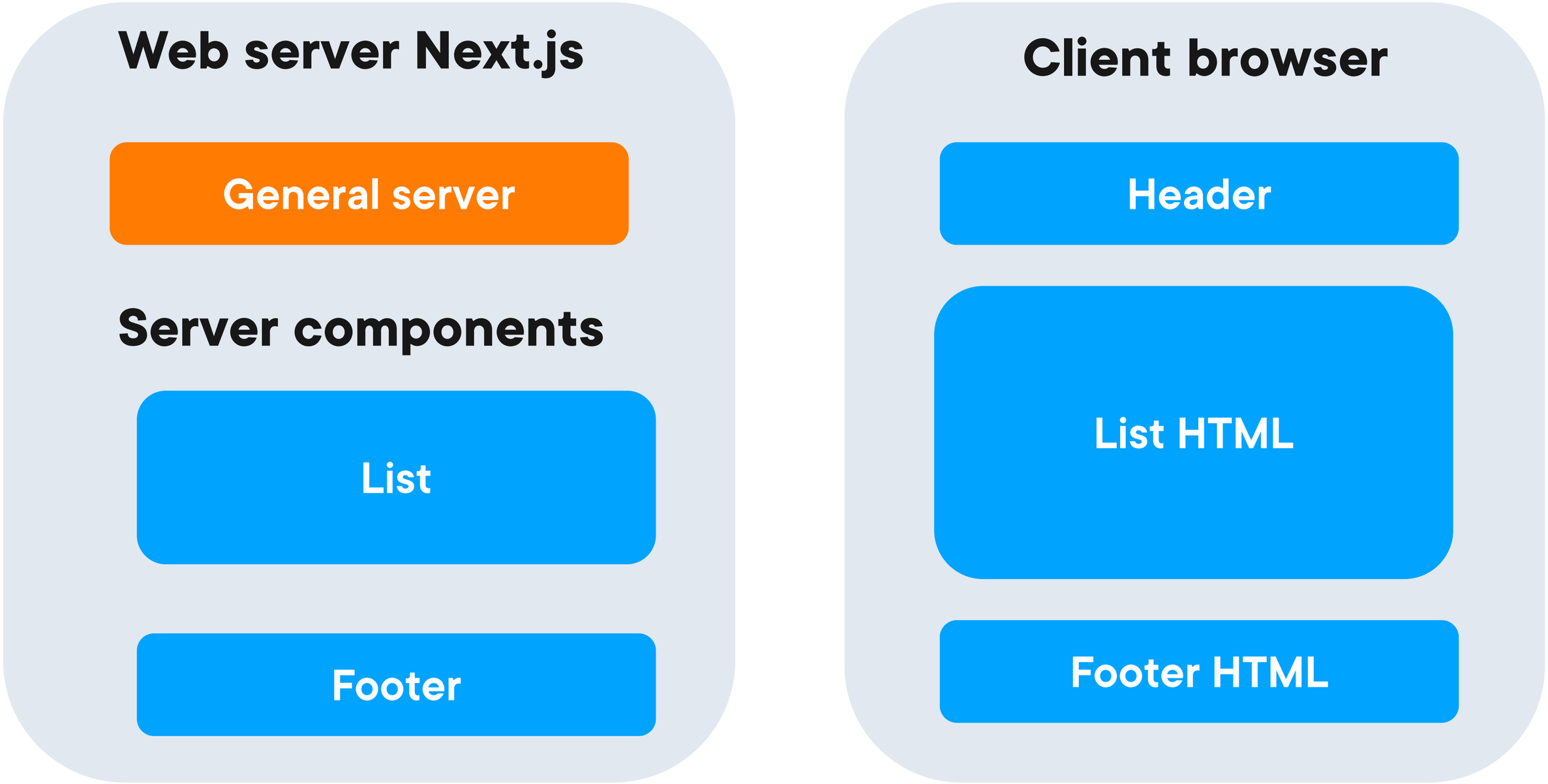

Web server Next.js
General server
Server components
List
Footer
Client browser
Header
@
@


Web server Next.js
General server
Server components
List
Footer
HTML
Client browser
Header
@
@

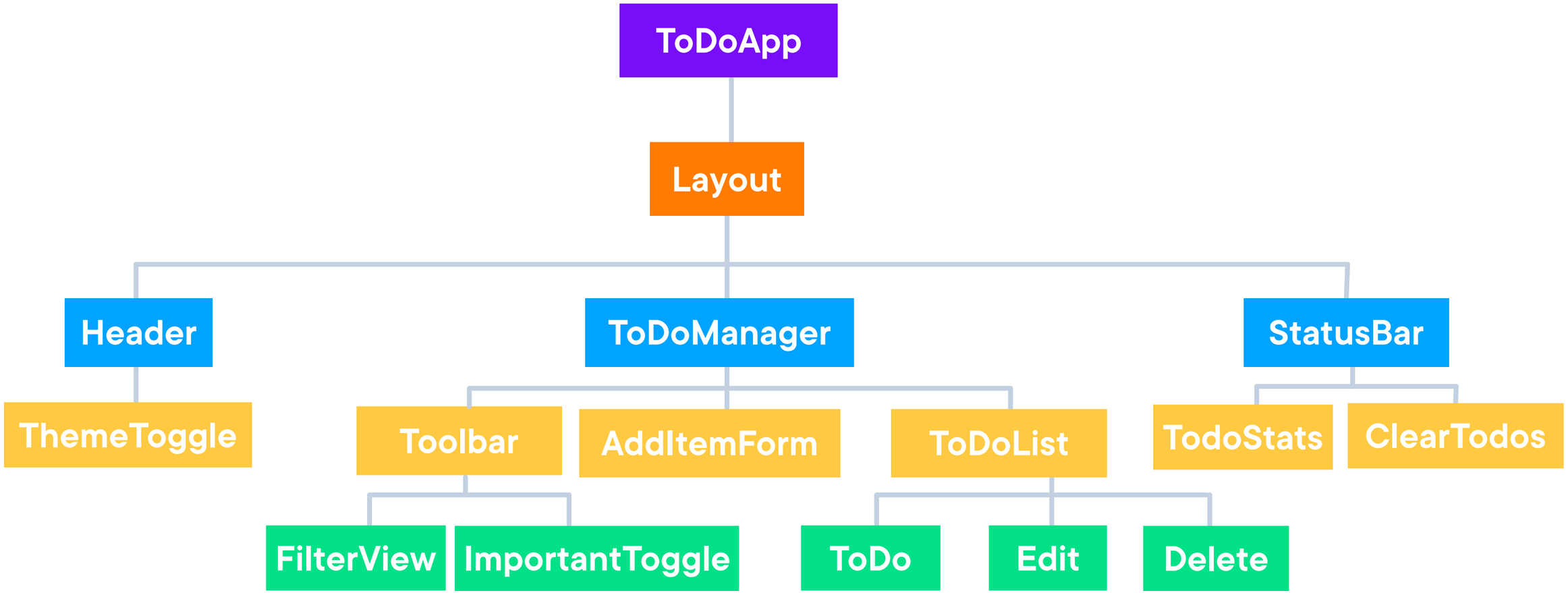

Web server Next.js
General server
Server components
List
Footer
HTML
HTML
Client browser
Header
@
@


Web server Next.js
General server
Server components
List
Footer
HTML
HTML
Client browser
Header
@
Footer HTML

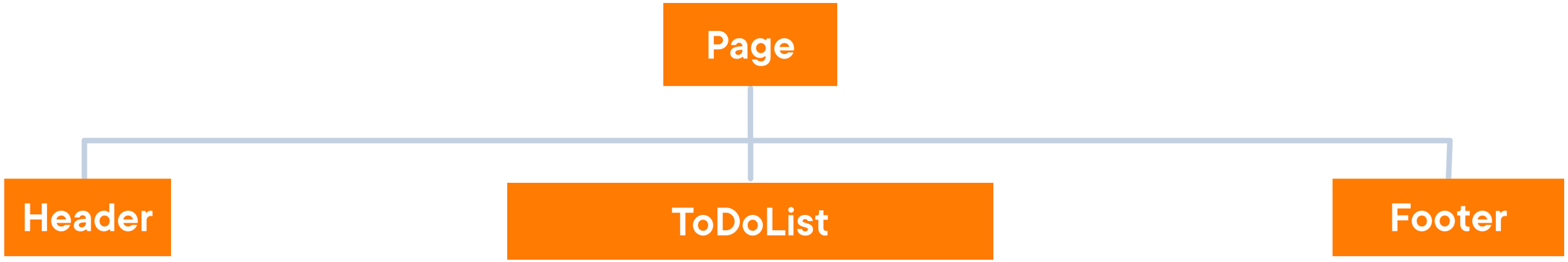

Web server Next.js
General server
Server components
List
Footer
HTML
HTML
Client browser
Header
List HTML
Footer HTML


Web server Next.js
General server
Server components
List
Footer
Client browser
Header
List HTML
Footer HTML

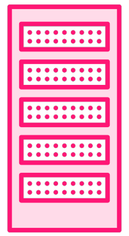
React Component Hierarchy (All Client Components)



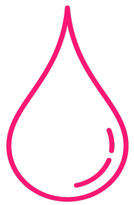
ToDo App Component Hierarchy (Server Components)



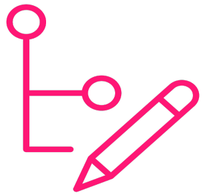
React Server Components and Events



Server Components can not process events

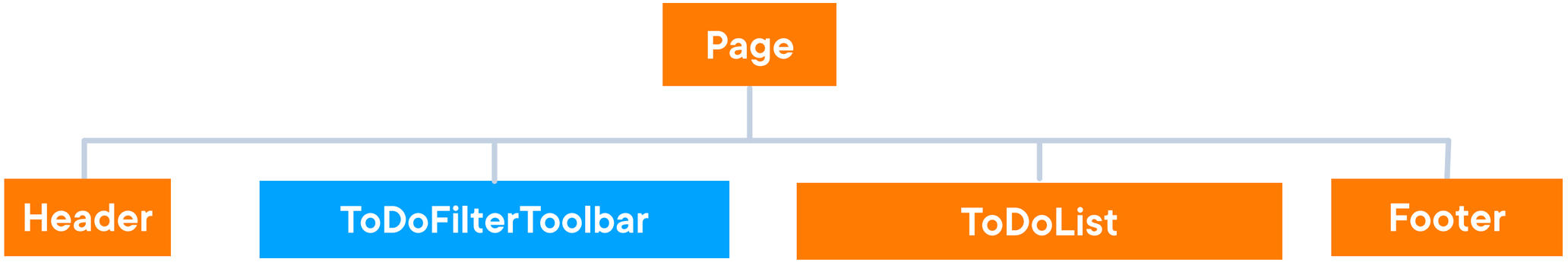


Server Components have no hydration steps

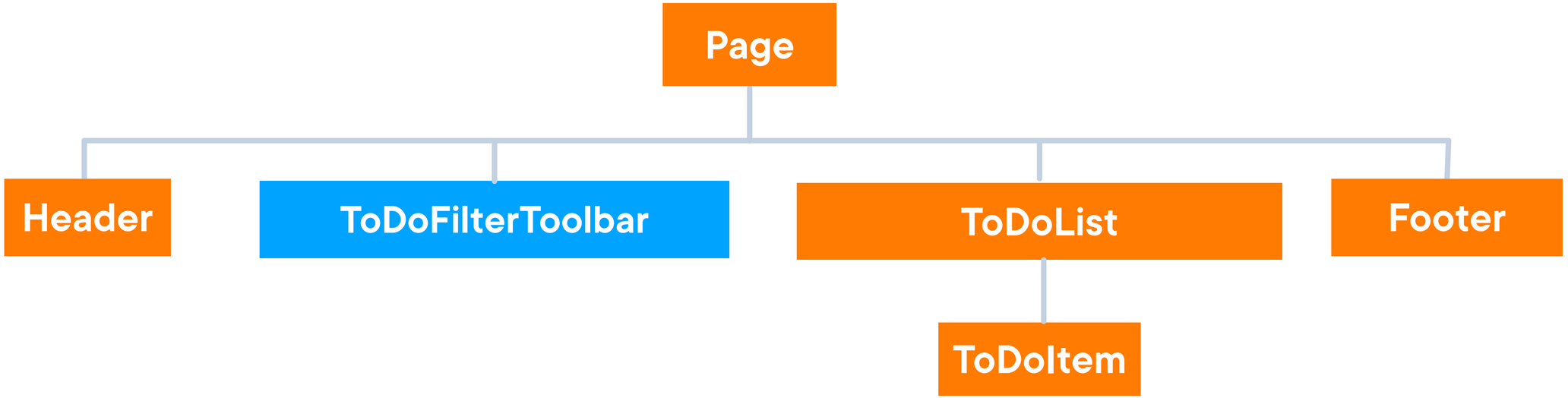


Client Components needed to add DOM events

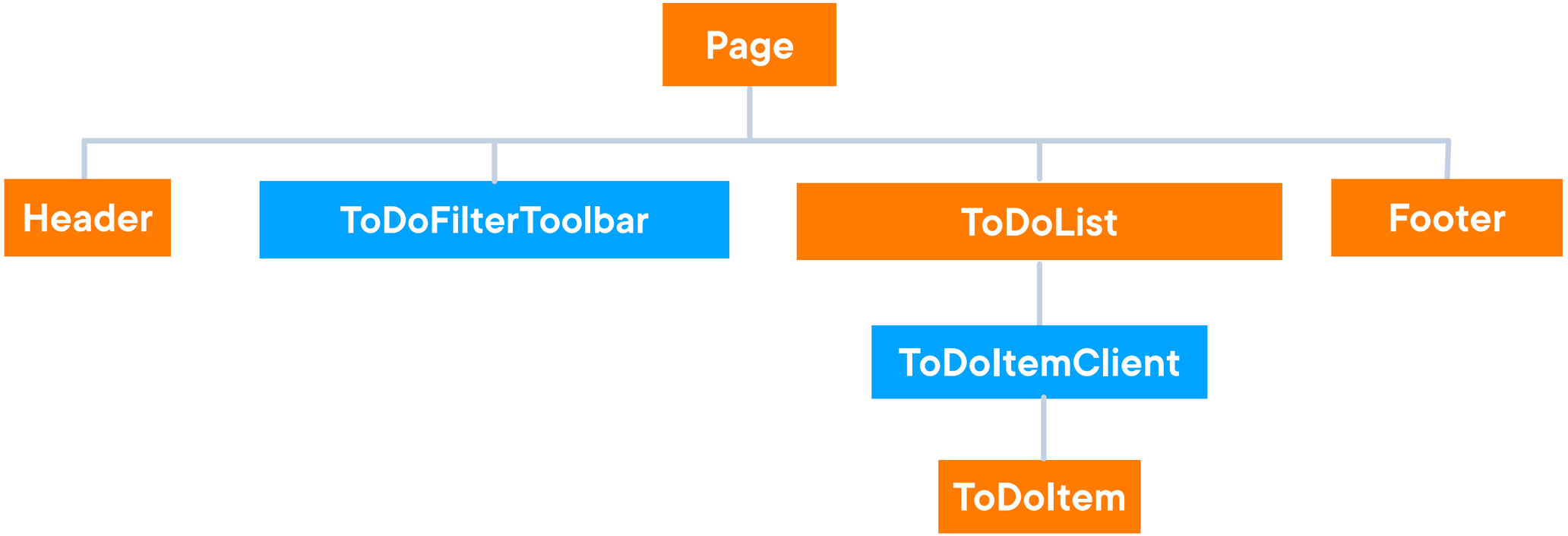
ToDo App Component Hierarchy (Server and Client Components)



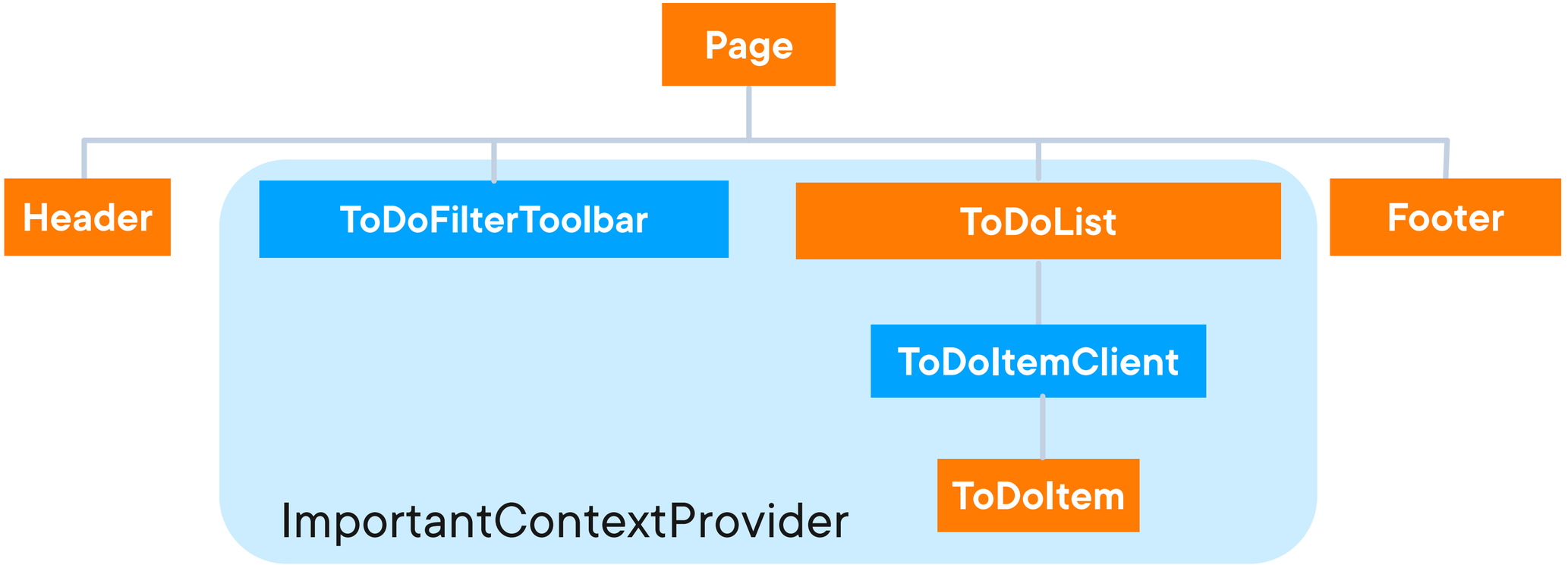
ToDo App Component Hierarchy (Server and Client Components)



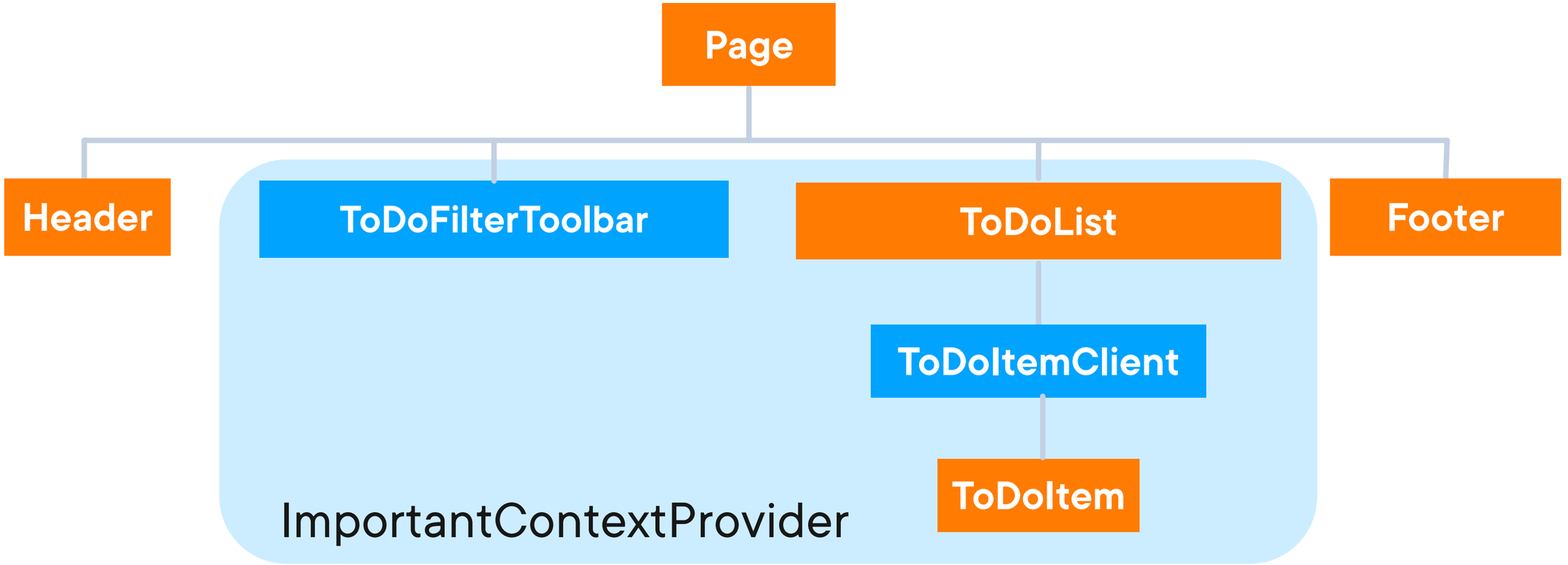
ToDo App Component Hierarchy (Server and Client Components)



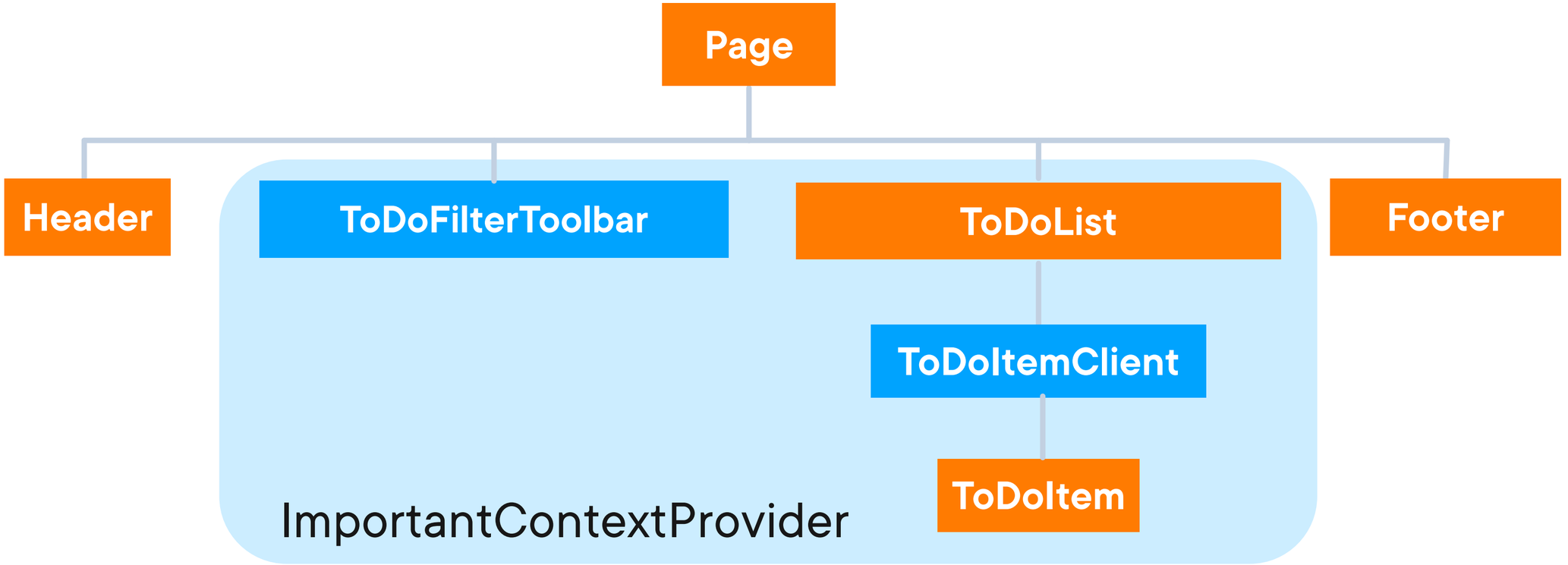
ToDo App Component Hierarchy (Server and Client Components)



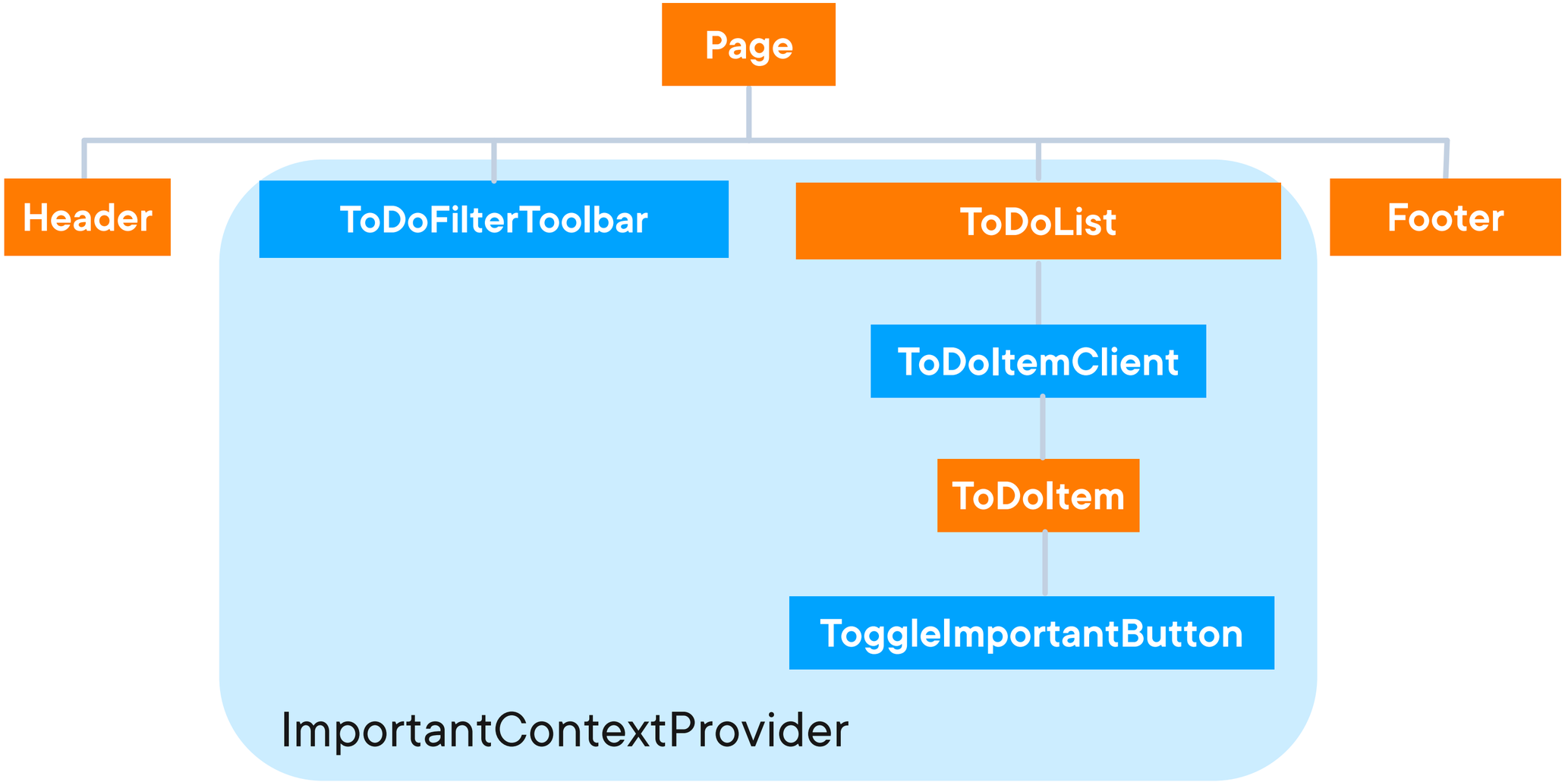
ToDo App Component Hierarchy (Server and Client Components)



ToDo App Component Hierarchy (Server and Client Components)



ToDo App Component Hierarchy (Server and Client Components)



Async Data Fetching in Server Components

How React Server Components can improve user browser experiences

How to build a React App that uses both server and client components

How React Server Components can improve user browser experiences

How to build a React App that uses both server and client components

We've not talked about coding experiences when building apps combing server and client components.

How React Server Components can improve user browser experiences

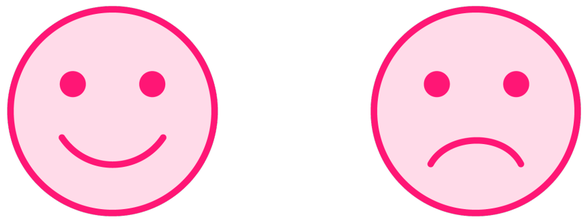
How to build a React App that uses both server and client components

We've not talked about coding experiences when building apps combing server and client components.

How React Server Components can improve user browser experiences

How to build a React App that uses both server and client components

We've not talked about coding experiences when building apps combing server and client components.



So Far...

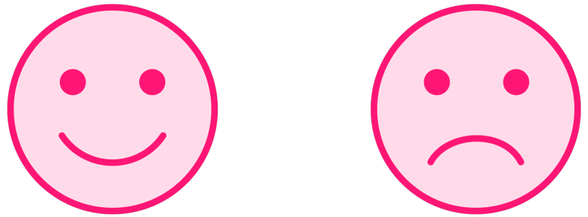
But

So Far...

How React Server Components can improve user browser experiences

How to build a React App that uses both server and client components

We've not talked about coding experiences when building apps combing server and client components.



Simple React Server Component

(no need to manage loading state)

ToDoList.js

```
async function ToDoList() {

  const response = await fetch('/api/todos’);
  const todosData = await response.json();

  return (
    <ul>
      {todosData.map((toDo) => {
        return <li key={toDo.id}>{toDo.text}</li>
      })}
    </ul>
  );
}
```

Page vs App Router in Next.js

Page Router

Built-in to Next.js since first release

Base components in /pages folder

/pages/index.js renders at http://localhost:3000/

/pages/customers.js renders at http://localhost:3000/customers

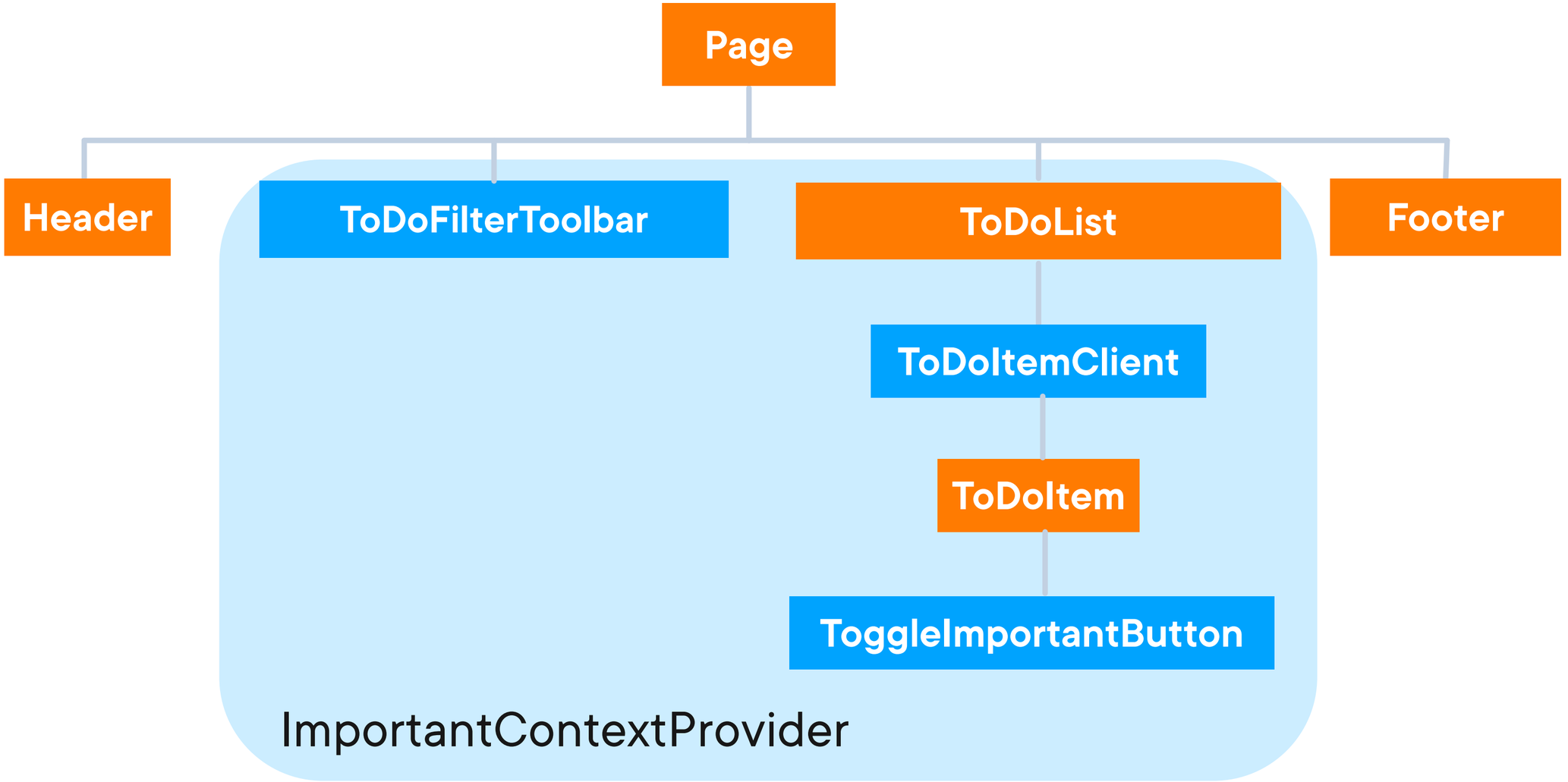
App Router

Optional router introduced in 2023

Base component is /app/page.js folder

/app/pages.js renders at http://localhost:3000/

ToDo App Component Hierarchy (Server and Client Components)



Takeaways

React Server Components evolutionary

React Concurrent Rendering and Streaming are enabling technology

React Server Components takes server-side rendering to the next level

Using Server Components in your apps is worth the investment

Check out Pluralsight course “Server Component Fundamentals in React 18”