Client and Server Routing

David Starr

@elegantcoder ElegantCode.com

Server Side Routing

Server Side Routing

Client makes request to:
//localhost/people/26

Server receives request

Server parses URL to see if it matches a configured route

Server executes code - logic ensues

Server replies with new page

Client receives response

```
app.all('*', express.static('public/index.html'));
```

Server Side Route Example

In Node.js

```
app.UseMvc(routes => {
  routes.MapRoute("default","{controller=Home}/{action=Index}/{id?}");
});
```

Server Side Route Example

In ASP.NET MVC with C#

Client Side Routing

Pro

Routing is generally faster

Smooth transitions are easy to implement

It's simple to render client-side views

Can render only part of a page without a full page refresh

No round trip to the server

Con

Often, the whole JavaScript-based web application is loaded all at once on the first request

Can be more complex

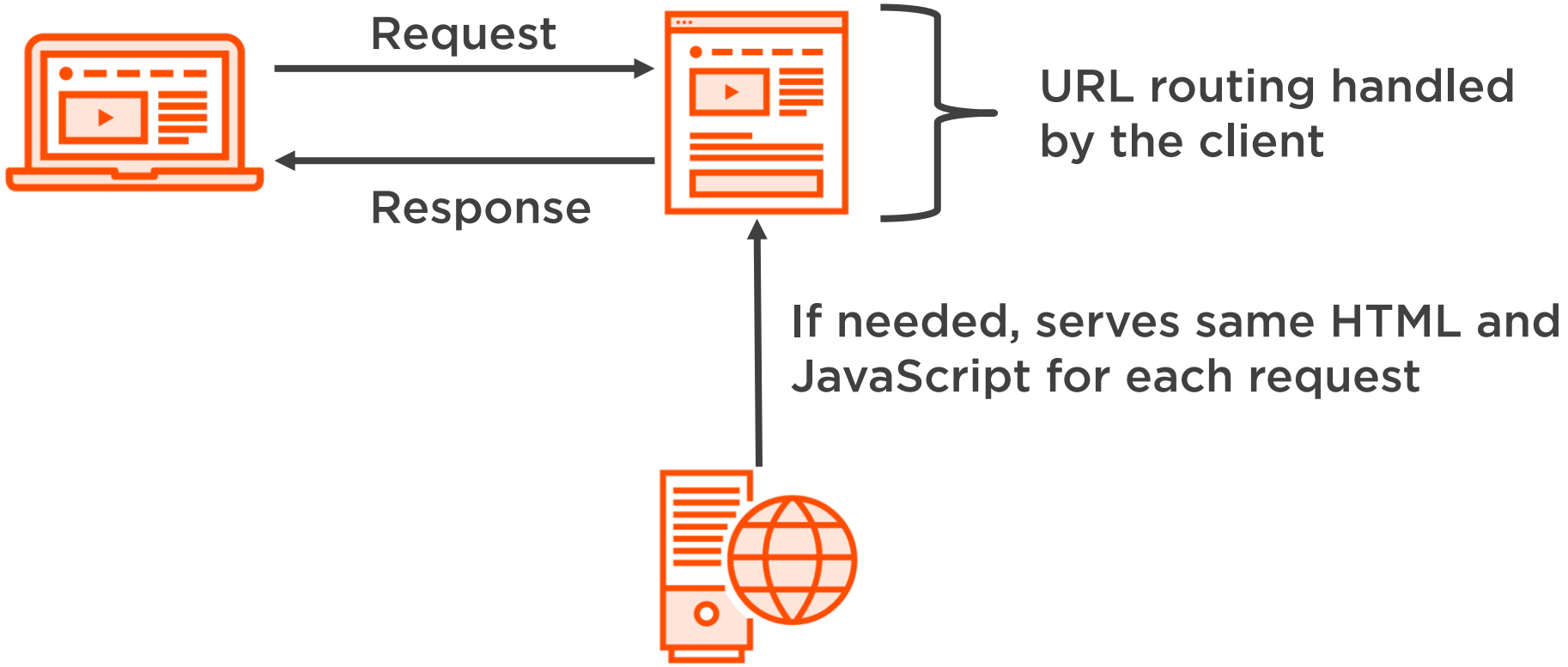
SEO may not work very well

May grow complex as routes are handled on both the client and the server

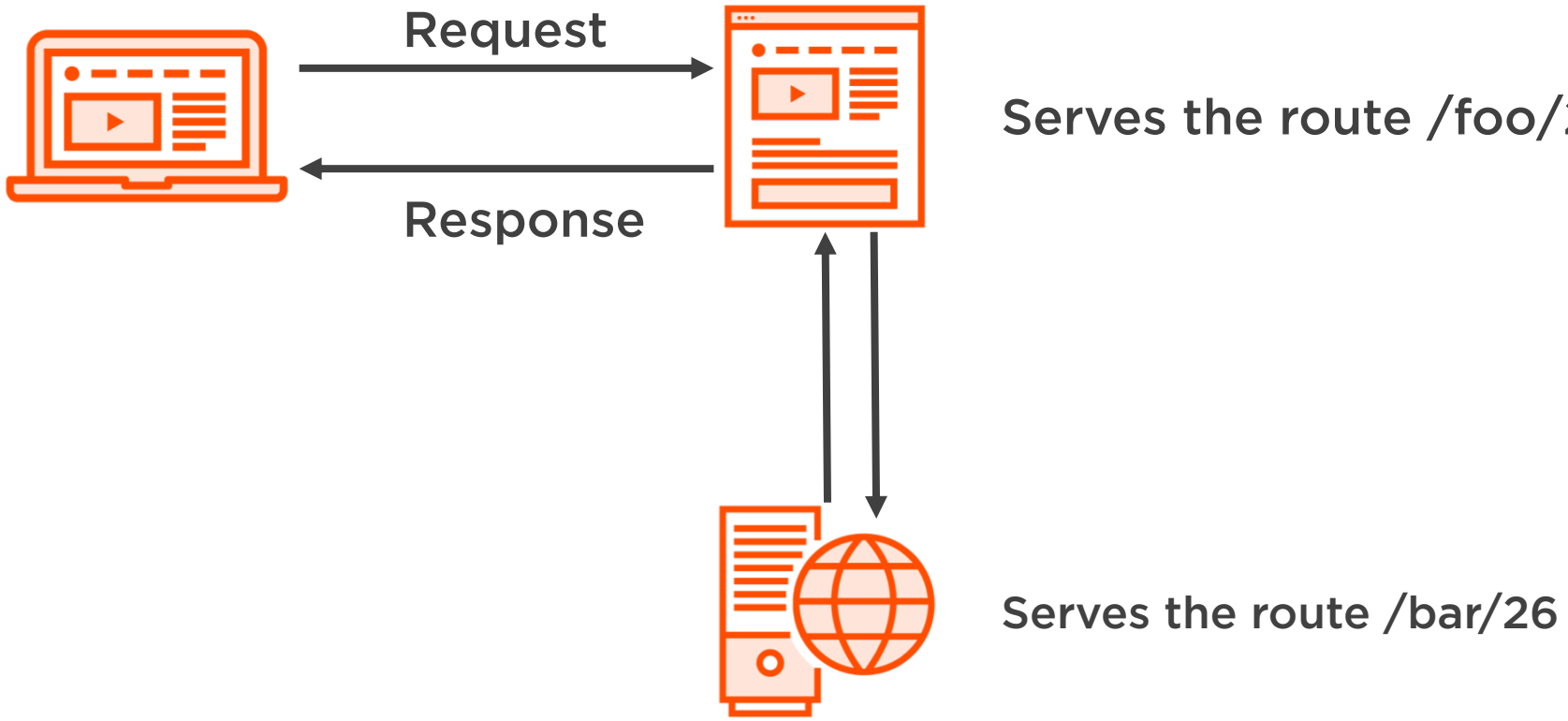
Demo

HashRouter vs. BrowserRouter

Client and Server Working Together



Client and Server Working Together



Summary

Server-side routing

Client-side routing

Server and client working together