Going to the SPA

A brief discussion on Client-Side Routing

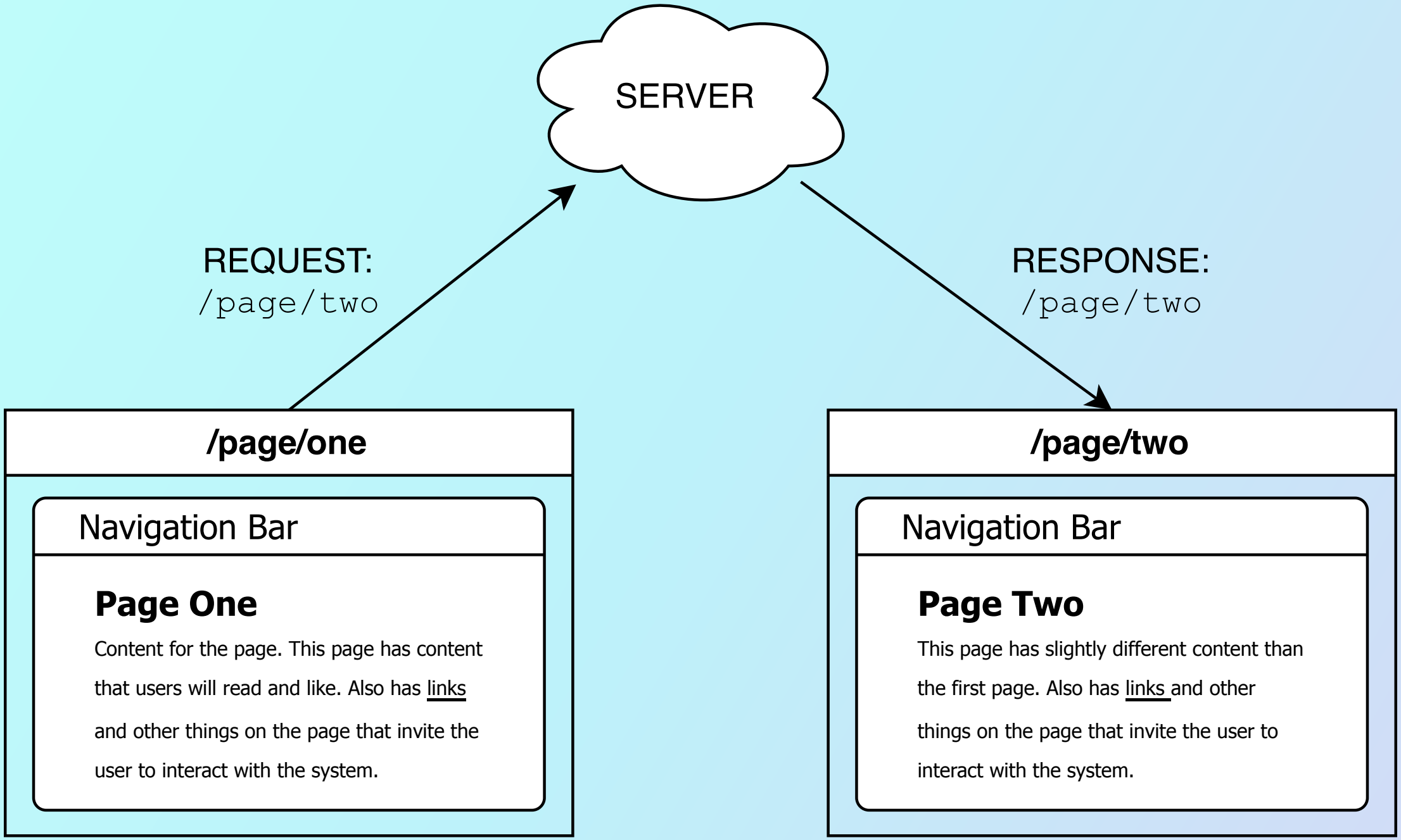
Navigation & Routing

```
<a href="/settings">
  Settings
</a>
```

GET request

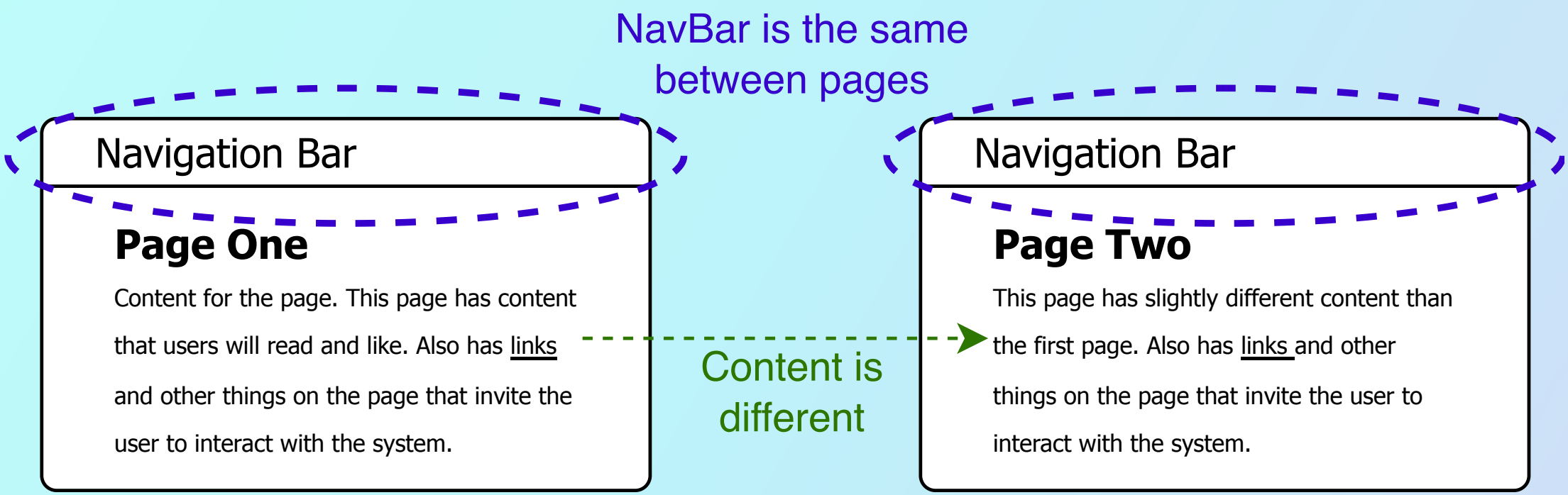
```
GET /settings HTTP/2
Host: example.org
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; fr; rv:1.9.2.8) Gecko/20100722 Firefox/3.6.8
Accept: */*
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Referer: http://example.org/test
Cookie: foo=bar; lorem=ipsum;
```

Server Response



Waterfall

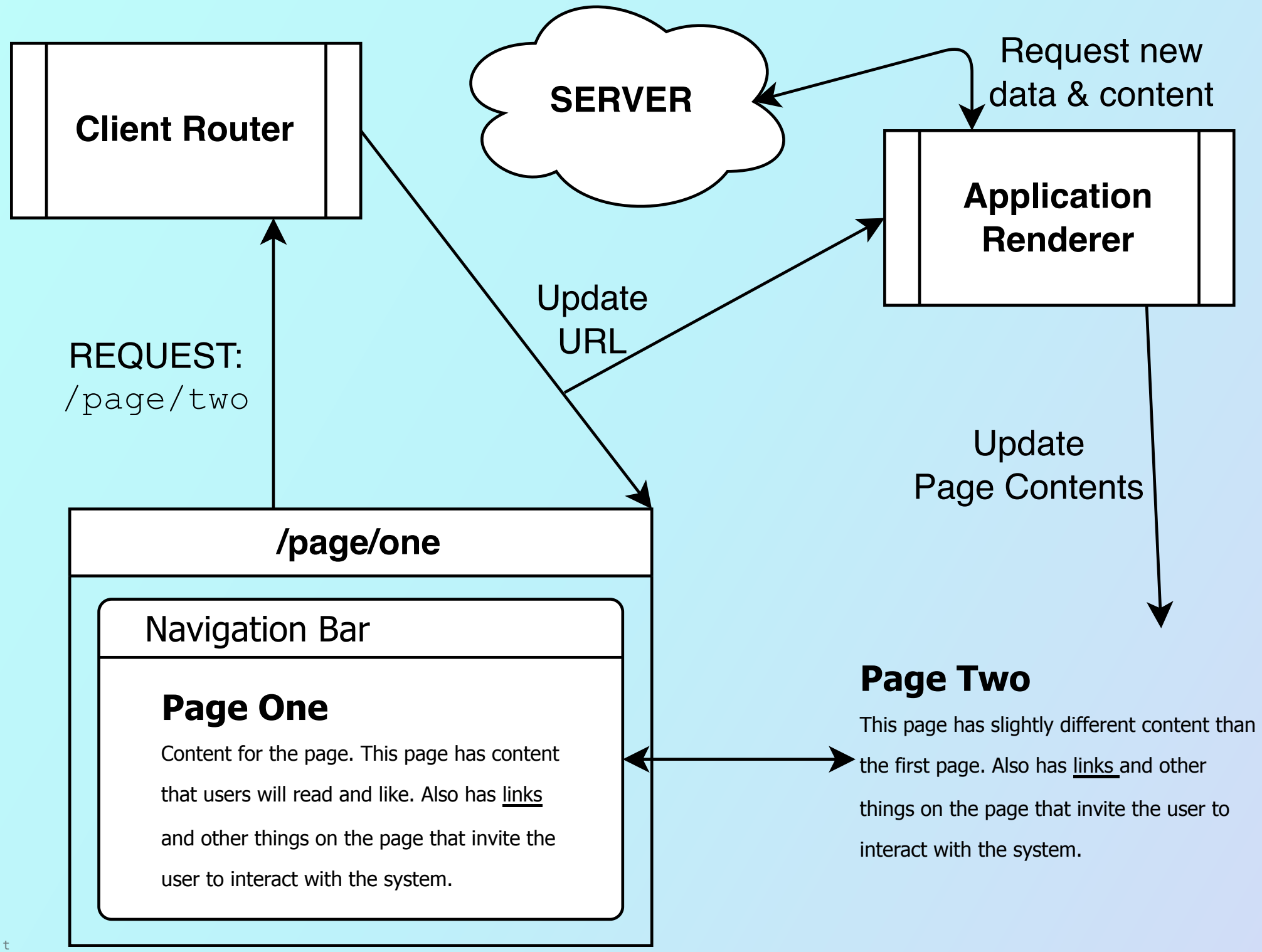
Server Response



Welcome to the SPA

(Single-Page Application)

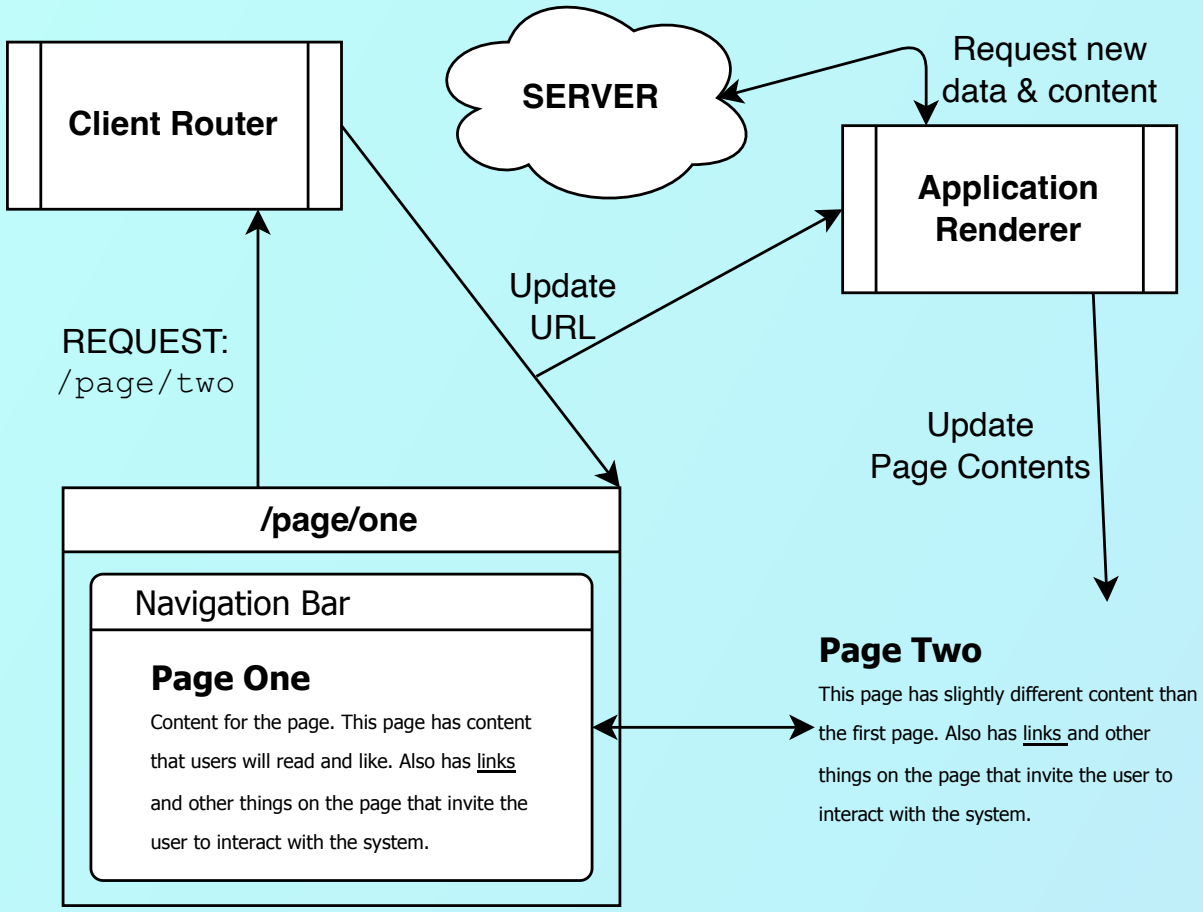
SPA Routing



SPA Routing

```
app.get('*', (req, res) =>
  res.render('index')
)
```

SPA Routing with React



- Application Renderer: `react`
- Client Router: `react-router-dom`
- *Request new data & content*: **our code**

React Router

```
<Router>
  <Switch>
    <Route path="/:id" Component={Page} />
    <Route path="/" Component={Landing} />
  </Switch>

<Link to="/">Home</Link>
```

History API

Manage navigation events in the browser

History API

`history.pushState()` allows us to **add an entry** into the browser tab's history stack. It takes three arguments:

- `state`: App defined state object (optional)
- `title`: Page title (obsolete but there for backwards compatibility)
- `path`: The new path to be displayed in the URL

```
history.pushState(
  { path: '/path' }, "Page Title", "/path"
)
```

History API

`history.replaceState()` allows us to **replace the current entry** in to the browser tab's history stack.

```
history.replaceState(
  { path: '/path' }, "Page Title", "/path"
)
```

Should I *push* or *replace*?

`pushState`

- adds an entry to the tab's navigation history
- good for handling user-driven navigation events (like a link click)

Example:

```
/tree/person/details/S0M3-P1D ==>
/tree/person/details/N3XT-P1D
```

`replaceState`

- overwrites the current history entry
- good for handling system-driven navigation events (like a redirect)

Example:

```
/tree/pedigree/S0M3-P1D ==>
/tree/pedigree/landscape/S0M3-P1D
```

Handling Navigation

```
function handleLinkClick (event) {
  if (!shouldHandleNav(event)) return

  event.preventDefault() // update the URL manually
  const { target: { href } } = event
  history.pushState({ path: href }, null, href)

  // notify application render state has changed
  handleNavigation({ path: href })
}

// handle all link clicks individually
anchor.addEventListener('click', handleLinkClick)

// alt option: handle all clicks with a universal
// handler (would need to check if target was a link)
window.addEventListener('click', handleLinkClick)
```

PopState Event

Fires when the user clicks the ← or → buttons, or when `history.go()` is called.

- `history.back()`
- `history.forward()` - yes, it fires a *pop* state event

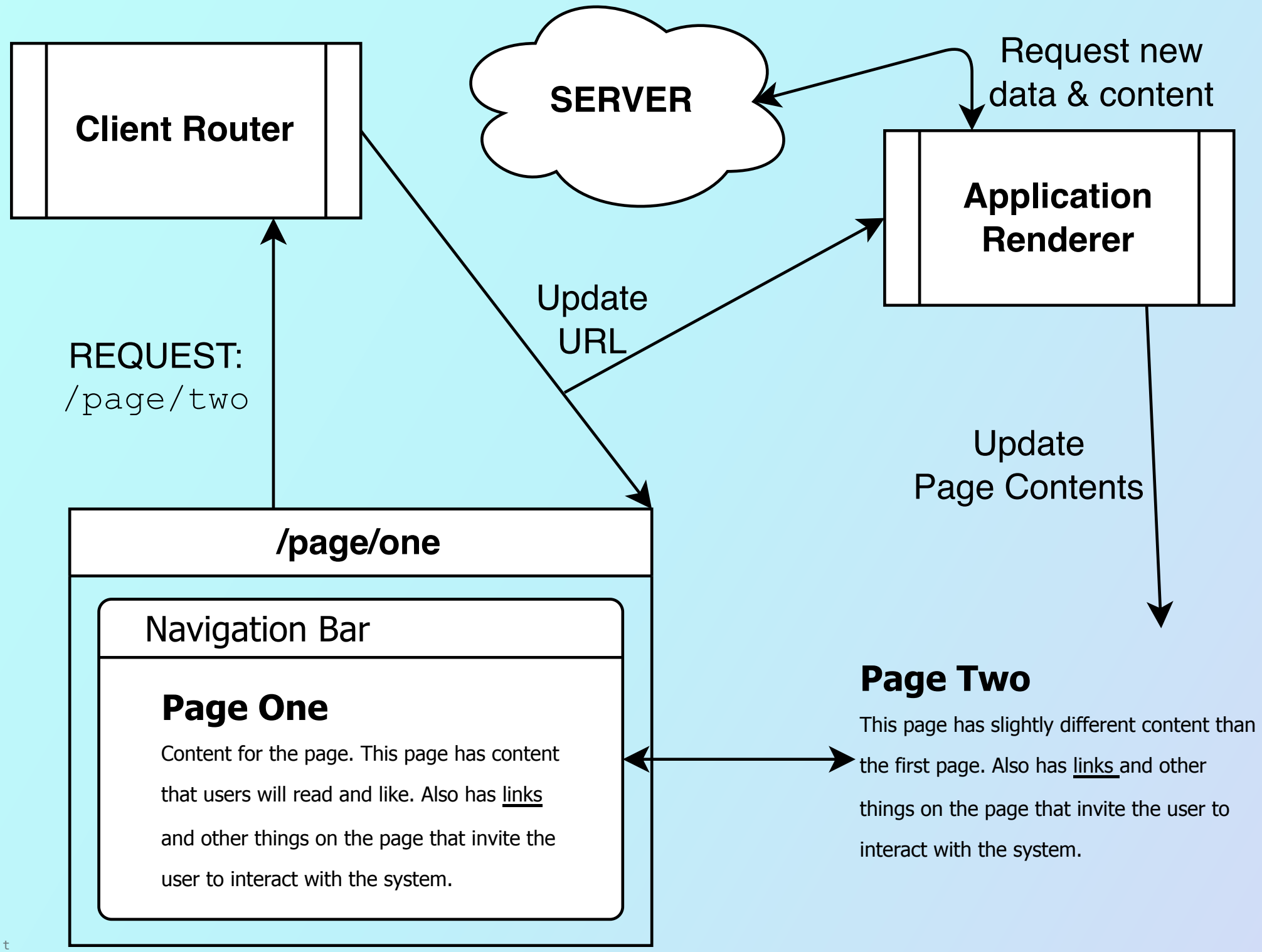
```
function handleStateChange (event) {
  // contains the contains the state object
  // we gave `history.pushState` earlier.
  const { state } = event
  handleNavigation(state)
}

window.addEventListener('popstate', handleStateChange)
```

So... what's `external` mean?

```
<Link
  to="/some/place"
  external
/>
```

SPA Routing



Questions?