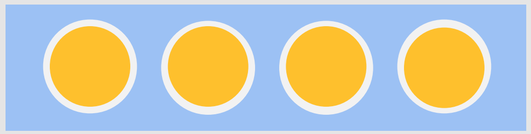
DEV ACADEMY
TE KURA HANGARAU
O AOTEAROA

Client-Side Routing *with* React Router

Serving a static HTML file

client

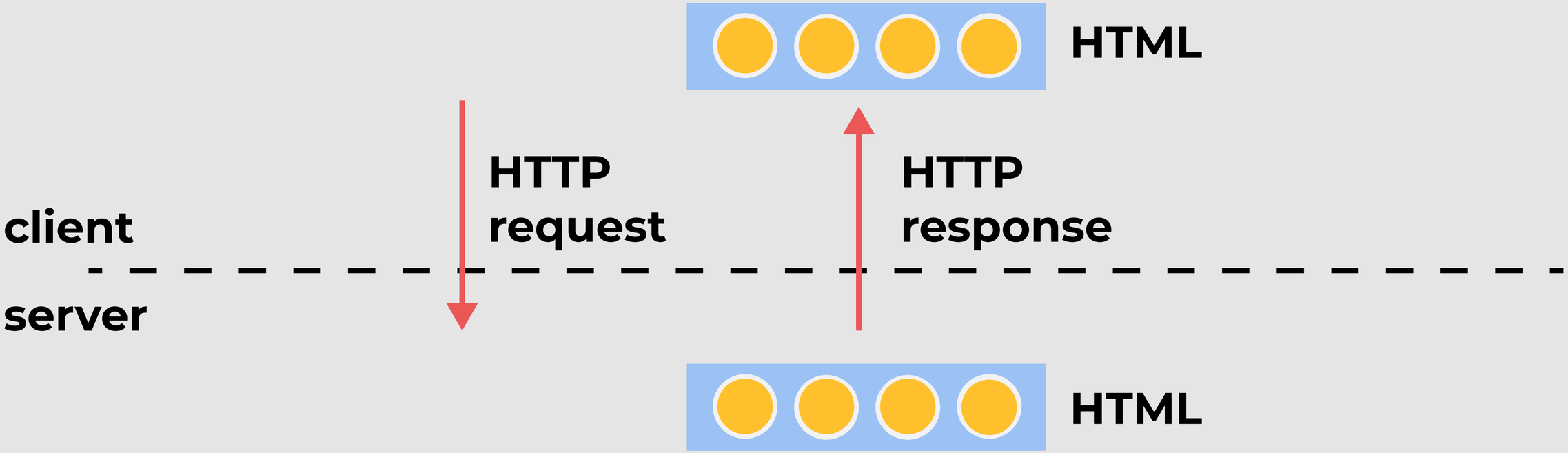
server



HTML

Serving a static HTML file

Serving a static HTML file



Serving a static HTML file

- We go to a URL like:
    - `http://localhost:3000/profiles/1`

```
server.get('/profiles/:id', (req, res) => {
  const id = req.params.id
  if (id === '1') {
    res.sendFile(path.resolve(__dirname, './public/silvia.html'))
  } else if (id === '2') {
    res.sendFile(path.resolve(__dirname, './public/sampson.html'))
  }
})
```

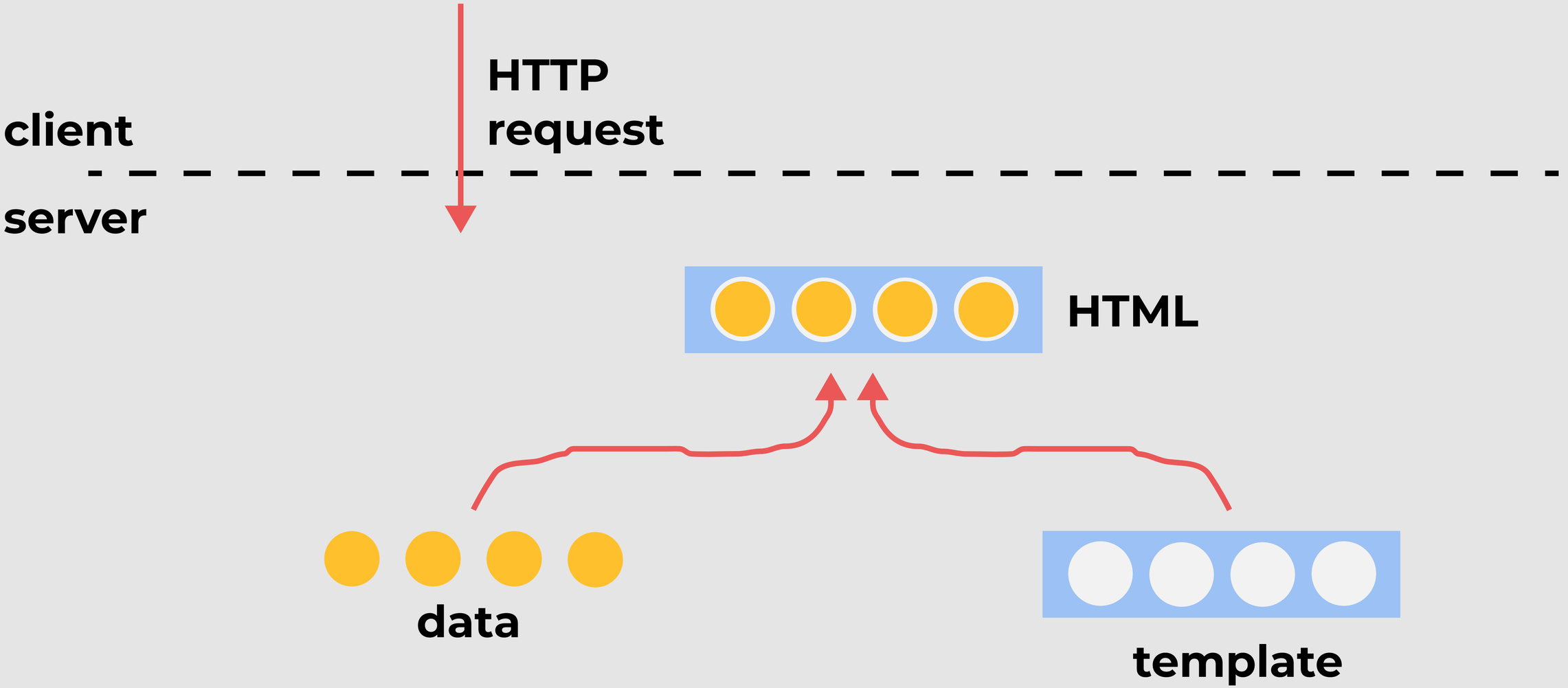
- We get static HTML file

Server-side rendering

Server-side routing

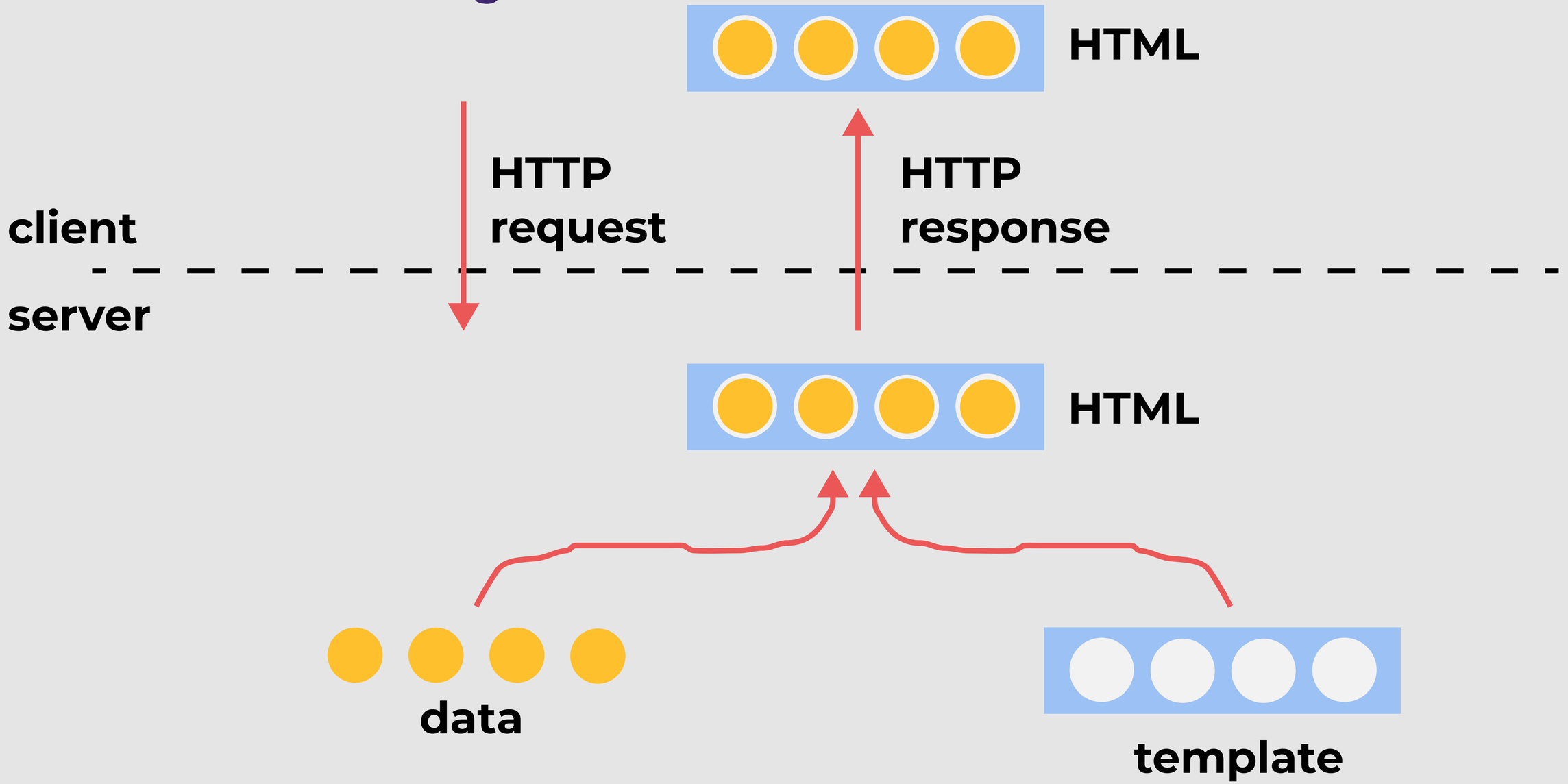
Server-side rendering

Server-side routing



Server-side rendering

Server-side routing



Server-side rendering

Server-side routing

- We go to a URL like:
  - `http://localhost:3000/locations`

```
// GET /locations
router.get('/', async (req, res) => {
  const locations = await db.getAllLocations()

  const viewData = { locations }
  res.render('showLocations', viewData)
})
```

`server/routes/locations.js`

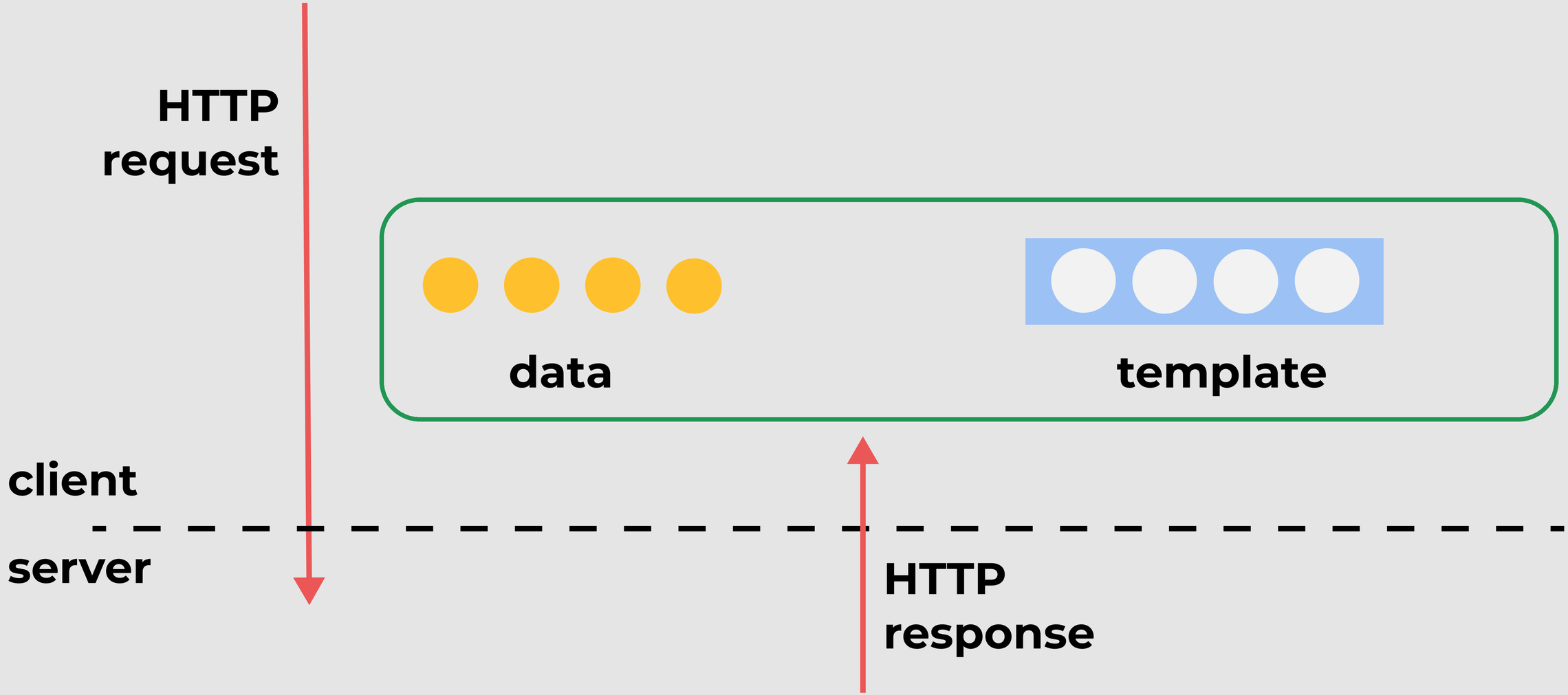
- Each new page required a new request and response

Client-side rendering

Client-side routing

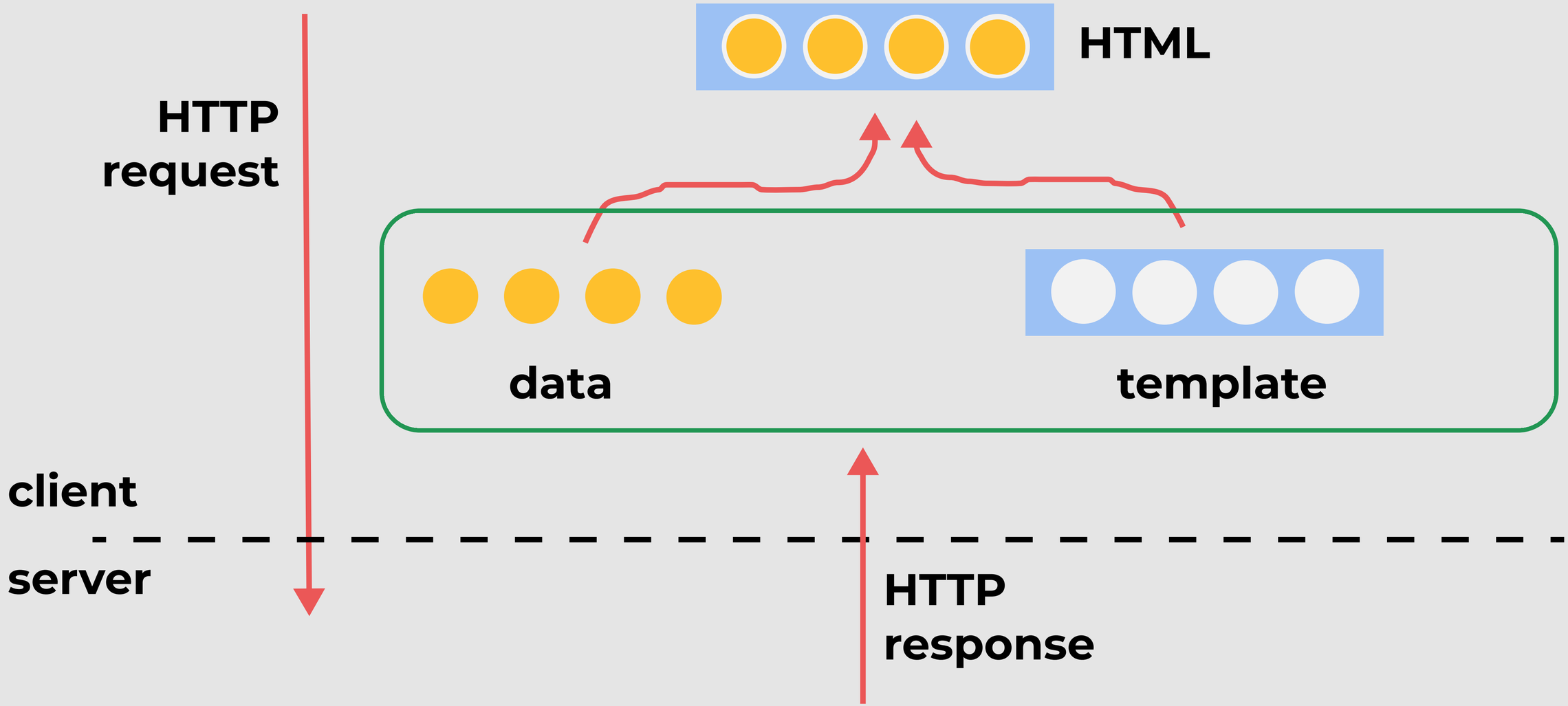
Client-side rendering

Client-side routing



Client-side rendering

Client-side routing



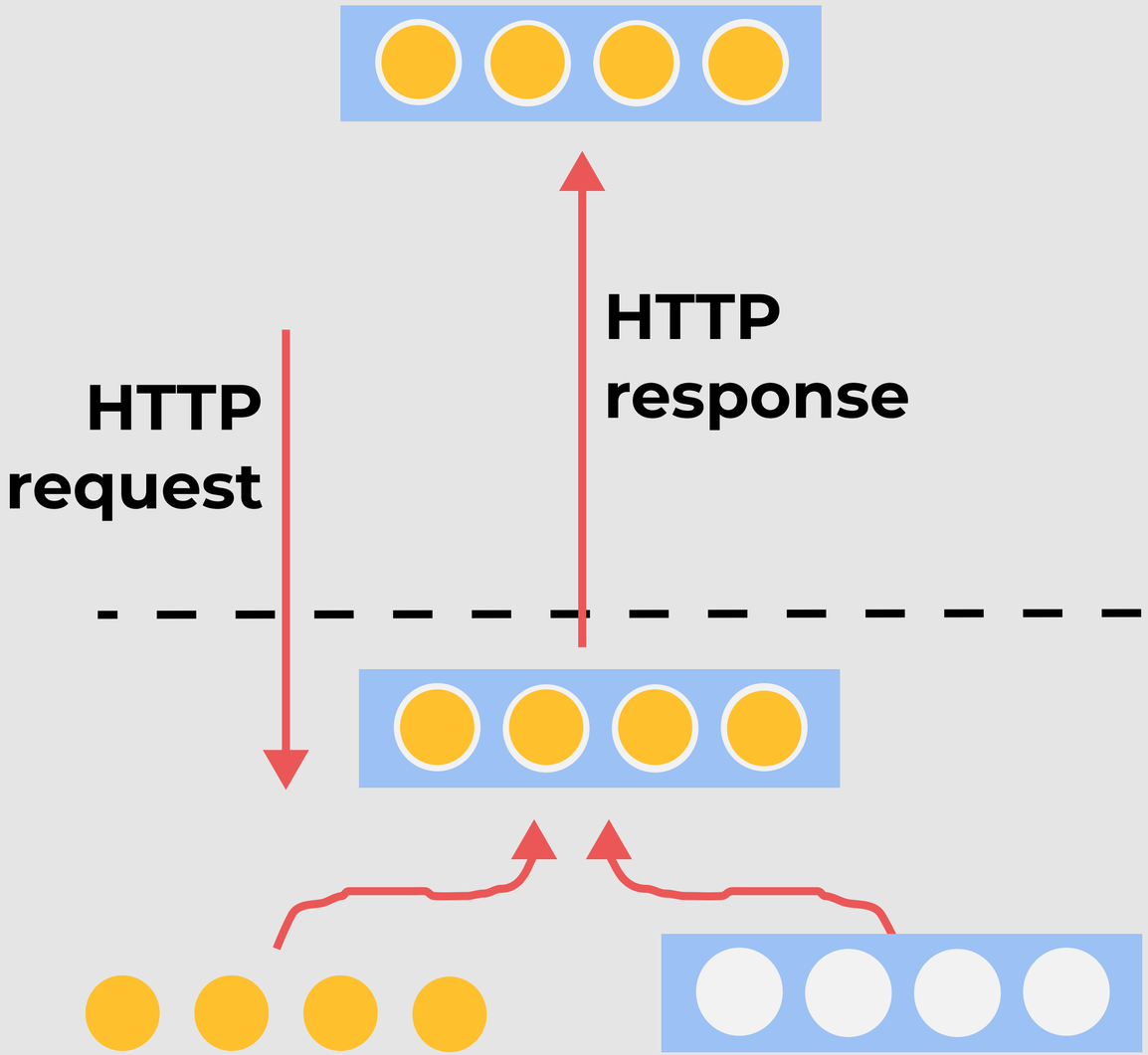
Client-side rendering

Client-side routing

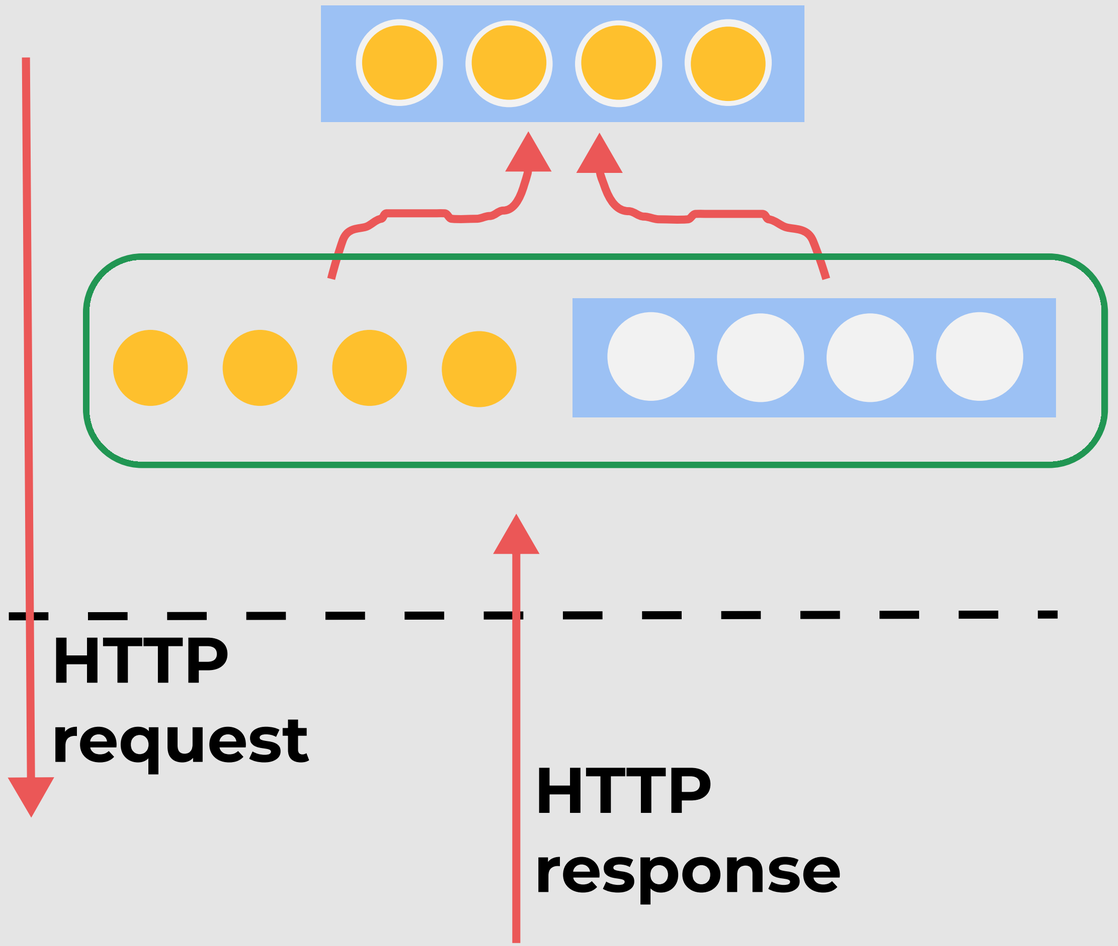
Navigating between pages happens entirely on client side:

- No new http request
- No http response
- All the templates (components) sent when the page first loads
- We’re just showing/hiding components
- Calls to server are (usually) only for data

Server-side rendering



Client-side rendering



Why use client-side rendering (vs server-side)?

All the reasons we use React...

- Responsive UI, fast loading between “pages”
- Processing work is on the client-side, where we don't have to pay for it
- Home computers (+ phones) are now fast enough

Client-side rendering

(without React Router)

```
export default function App() {
  const [page, setPage] = useState('home')

  switch(page) {
    case 'home':
      return <Home />
    case 'info':
      return <Info />
    case 'cats':
      return <Cats />
  }
}
```

Client-side rendering

(without React Router)

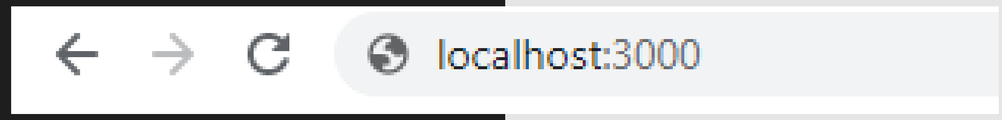
```
export default function App() {
  const [page, setPage] = useState('home')

  switch(page) {
    case 'home':
      return <Home />
    case 'info':
      return <Info />
    case 'cats':
      return <Cats />
  }
}
```

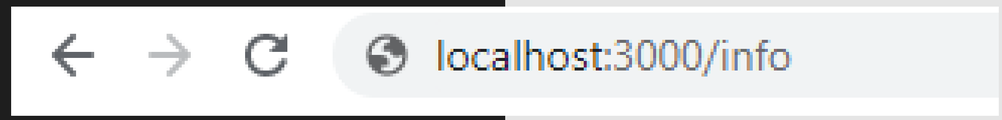
- URL always stays the same
  - Cannot share URL links
  - Will always start at homepage
- No browser history
  - No back button
  - No forward button
- We'll need to be able to call setPage everywhere in our application

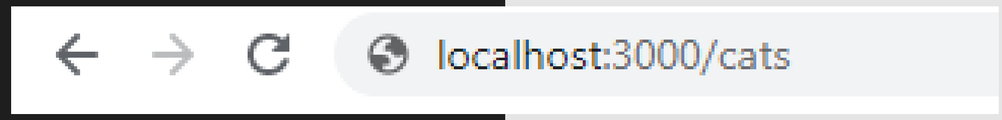
Client-side routing - sneak preview

With React Router

```jsx
<Route path='/'>

  <Route index element={<Home />} />

  <Route path='info' element={<Info />} />

  <Route path='cats' element={<Cats />} />

</Route>
```

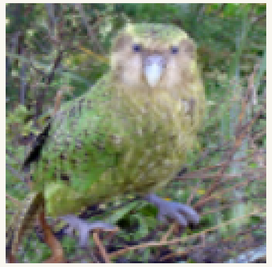






React Router docs

https://reactrouter.com/en/main



Birds of Aotearoa

localhost:3000/birdFamily/Parrots

host port path

My favourite native birds

Nav

- Info
- Parrots
- Honeyeaters
- Pigeons
- Rails and swamphens
- Wattlebirds
- Owls

Parrots

Code demo...

Steps for using React Router

1. Create the component
2. Add the <Route>
3. Build the <Link>
4. (if required) make the component read the params from the URL

Client-side routing with React Router: Summary (1)

- More responsive user experience than server-side routes
- Allows sharing of URLs
- Allows you to use the Back and Forwards buttons

- **<Route>** - connects URL paths to components
- **<Link>** - how you navigate between <Route>s

Client-side routing with React Router: Summary (2)

- Renders the most specific matching route (instead of first matching route)
- Using map to render elements multiple times:

```
{birdFamilies.map(birdFamily => <li>{birdFamily}</li>)}
```

- String interpolation:

```
<Link to={`/birdFamily/${birdFamily}`}>{birdFamily}</Link>
```

- Route params

```
<Route path='birdFamily/:birdFamily' element={<BirdFamily/>} />
```

```
function BirdFamily () {
  const params = useParams()
  const birdFamily = params.birdFamily
```