Kentico

Lecture 8

Petr Svirák

Routing in general

HTTP

- Plain ASCII text (messages) sent between a client and a server
  - initiated by client (browser) – request

```
POST /form.html HTTP/1.1
Host: localhost:3000

{ "email": "abc@mno.xyz"}
```

  - provides server data – response

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-length: 23
Content-type: application/json;
charset=utf-8
Date: Wed, 24 Oct 2018 20:15:46 GMT
etag →W/"17-Z51rgDtEQ9F6PHZZ4zRl8FfQaL8“

{ "id": “123", "email": "abc@mno.xyz" }
```

URI

- Unique Resource Identifier
- URI and URL often used interchangeably, URN almost unused
- URL breakdown:

```
        userinfo     host       port
         ┌───────┐ ┌──────────┐ ┌──┐
https://john.doe@www.example.com:123/forum/questions/?tag=networking&order=newest#top
└───┘ └──────────────────────────┘└────────────────┘└──────────────────────────────┘└─┘
  │            │                     │                          │                    │
scheme      authority               path                       query              fragment
```

Server-side Routing

- Default routing available
- Causes complete page refresh
- SEO friendly

```
https://www.example.com:123/forum/questions/?tag=networking&order=newest#top
└───┘└──────────────────┘└──┘└──────────────┘└────────────────────────────┘└──┘
  │            │           │          │                    │                │
scheme       host        port       path                query            fragment
```

- Scheme –> protocol
- Host –> DNS
- Port, Path –> server
- Path, Query –> server-side application
- Query, Fragment –> client-side application (JS or browser (anchors))

Client-side Routing (before HTML5)

- Limited by API (query was „readonly“)
- Fragments are not sent to server
- Some browsers hide query and/or fragments (suboptimal for URL slugs)
- SEO unfriendly (fragments are ignored, parameters in query must often be explicitly listed)

```
https://www.example.com:123/forum/questions/?tag=networking&order=newest#top
└───┘└──────────────────┘└──┘└──────────────┘└────────────────────────────┘└─┘
  │           │           │          │                      │               │
scheme      host        port       path                   query           fragment
```

- Scheme –> protocol
- Host –> DNS
- Port, Path –> server
- Path, Query –> server-side application
- Query, Fragment –> client-side application (JS or browser (anchors))

Client-side Routing (after HTML5)

- HTML5 added history.pushState, history.replaceState and the onpopstate event
- (almost) as SEO friendly as server-side routing
- Social networks often require a server-side middleware to get meta-tags right
- Server needs to serve or redirect to client's code (SPA) from unknown routes

```
https://www.example.com:123/forum/questions/?tag=networking&order=newest#top
└───┘└──────────────────┘└──┘└──────────────┘└────────────────────────────┘└─┘
  |          |            |          |                    |                |
scheme      host         port       path                query            fragment
```

- Scheme –> protocol
- Host –> DNS
- Port, Path –> server
- Path, Query –> client-side application (JS)
- Fragments –> client-side application (mostly browser (anchors))

React router

Routers

- Wrapper component around the app (similar to react-redux's <Provider>
- What type of routes is used

- StaticRouter –> server-side rendering and static routing
  https://www.fencyDomain.rip/some-custom-routes/here/and/there
- HashRouter –> client-side routing that only uses fragment part of URL
  https://www.fencyDomain.rip/#some-custom-routes/here/and%23there
- **BrowserRouter** –> leverages HTML5's history API to work URL in browser
  https://www.fencyDomain.rip/some-custom-routes/here/and#there
- NativeRouter –> react-native routing (in mobile apps)
- MemoryRouter –> stores URL in memory only (test, apps lacking adress bar)

Kentico

Link components

- anchors (<a/>) –> <Link /> components
- prevent roundtrip to the server
- props
  - to – relative path that always begin at application root level (“/”)
  - replace – instead of add up to (browser’s) history, clicking link just replaces last record
- NavLink extends link by allowing special style/class to be added when link matches location
  - Cannot be used with Bootstrap since activity of a navigation link is not set at anchor tag level

Kentico

Static routing

- Routes are defined at one place and prior application start
- Similar to (and typical for) server-side applications
  - Rails, Express, Ember, Angular, MVC
  - react-router v3 and earlier
- Since routes form a (static) hierarchy, component design bends to the very same pattern
- Components need to know they are routed (must render children)
- Route components are not true components
  - only other Route children are allowed and they never render themselves

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

https://www.fencyDomain.rip/

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

```
<App>
    <Home />
</App>
```

https://www.fencyDomain.rip/about

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

```
<App>
  <About />
</App>
```

Kentico

Static routing example

https://www.fencyDomain.rip/picture

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

```
<App>
  <Profile>
    <Picture />
  </Profile>
</App>
```

Static routing example

https://www.fencyDomain.rip/inbox/messages
vs.
https://www.fencyDomain.rip/inbox/settings

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

```
<Route path="/" component={App}>
  <IndexRoute component={Home} />
  <Route path="about" component={About} />
  <Route path="inbox" component={Inbox}>
    <Route path="messages" component={Messages} />
    <Route path="settings" component={Settings} />
  </Route>
  <Route component={Profile}>
    <Route path="picture" component={Picture} />
  </Route>
</Route>
```

```
<App>
  <Inbox>
    <Messages />
  </Inbox>
</App>
```

```
<App>
  <Inbox>
    <Settings />
  </Inbox>
</App>
```

Kentico

Dynamic routing

- Starting with react-router v4
- Route evaluates during render phase (on the fly)
  - when route matches, specified component is rendered
  - when route does not match, null is rendered
- Declarativity supports versatility
  - Responsive routing (in combination with media query)
  - Conditional routing (based on user data – e.g. permission, time-bound, …)
  - Headings on page as sub-routes
  - Recursive routing
- Allows inclusive routing where multiple (sub)routes get matched to single path

Route component

- Props
  - path – string that has to match with current location (URL)
    might include paramters (e.g. „/thread/:threadId/:selectedCommentId?“)
    https://www.npmjs.com/package/path-to-regexp is used for matching
  - exact – matches only if location is exactly same as path (stops inclusion)
  - render * – invoked only when path matches (match routeProp is never null)
  - component * – renders only when path matches (match routeProp is never null)
  - children * – renders always (if path does not match, match routeProp is null)

- Rendering method (marked with *) are all provided with set of routeProps that include:
  - match – if not null, contains details on matched path
    (e.g. Route parameter or whether the match is exact)
  - location&history – https://reacttraining.com/react-router/web/api/history/history-is-mutable

Tag: router-06-inclusive-routing

Inclusive rendering

```
<div>
  <Route path="/" component={() => <h1>Home</h1>} />
  <Route path="/api" component={() => <h2>Api</h2>} />
  <Route path="/api/call" component={() => <h3>Call</h3>} />
  <Route path="/api/call/:number" component={({match}) => <h4>{match.params.number}</h4>} />
</div>
```

- Inclusive (by default) –> any match renders, including “subpaths”
- Example: for “/api/call/maybe-not” renders:

```
<div>
  <h1>Home</h1>
  <h2>Api</h2>
  <h3>Call</h3>
  <h4>maybe-not</h4>
</div>
```

Tag: router-07-exclusive-routing

Exclusive rendering

```
<Switch>
  <Route path="/" component={() => <h1>Home</h1>} />
  <Route path="/api" component={() => <h2>Api</h2>} />
  <Route path="/api/call" component={() => <h3>Call</h3>} />
  <Route path="/api/call/:number" component={({match}) => <h4>{match.params.number}</h4>} />
</Switch>
```

- Exclusive within Switch – only first match renders
- Example: for “/api/call/me-maybe” renders:

```
<h1>Home</h1>
```

- Example: for “/api/call/me-maybe”, if all routes were **exact**, renders:

```
<h4>me-maybe</h4>
```

Kentico

Parameters

(short demo)

Tag: router-08-redirect-parameters-declarative-routing

Recursive routes

(short demo)

Tag: router-09-recusrive-routing

Redirect component

- Props
  - to – where should be location redirected (equivalent of Link's to)
  - push – if true, redirect add new history entry (opposite of Link's replace)
  - from – has to match with current location for redirect to trigger (equivalent of Route's path)
  - exact – equivalent of Route's exact

- Can be used with e.g. access-controlled resources or responsive routing
- Can pass parameters *from* one route *to* another

Tags: router-12-withRouter & router-13-with-redux

withRouter & redux

- withRouter is equivalent of using a component in a route:
  <Route component={Component} />

- Every change in URL causes re-render of the component

- The connect result wrapped in withRouter effectively adds RouteComponentProps to wrapped each container.
  - But unless the history gets explicitly passed to a (thunk) action, actions cannot result in redirect
  - Also, replaying actions does not take effect on routing thus is easily becomes useless
  - https://github.com/supasate/connected-react-router to the rescue

Excercise

- Open Lecture8 folder in IDE
- Install packages (`npm install --no-optional`) & Start app (`npm run start`)

- Task 1
  - Assignment tag: router-task-1
  - Solution tag: router-solution-1
  - Show the "NotFound" component anytime an unknown route is matched in Content component

- Task 2
  - Assignment tag: router-task-2
  - Solution tag: router-solution-2
  - Allow only authenticated users to access Profile component.
    - Anonymous users should be redirected to /Auth route.
    - Use authenticationStore.isAuthenticated to determine when user is authorized
  - Use Docs: https://reacttraining.com/react-router/web/guides/quick-start
  - Attend to all TODOs in the code (there are 4 places requiring your attention)

Kentico

Other sources

- Demo notes

- Code examples
  - Tag: router-10-breadcrumb
  - Tag: router-14-without-unnecessary-rerenders
  - Tag: router-15-all-examples-enabled

- Read through official documentation:
  - https://reacttraining.com/react-router/web/guides/philosophy
  - https://reacttraining.com/react-router/web/guides/quick-start
  - https://reacttraining.com/react-router/web/example/preventing-transitions
  - https://reacttraining.com/react-router/web/example/auth-workflow