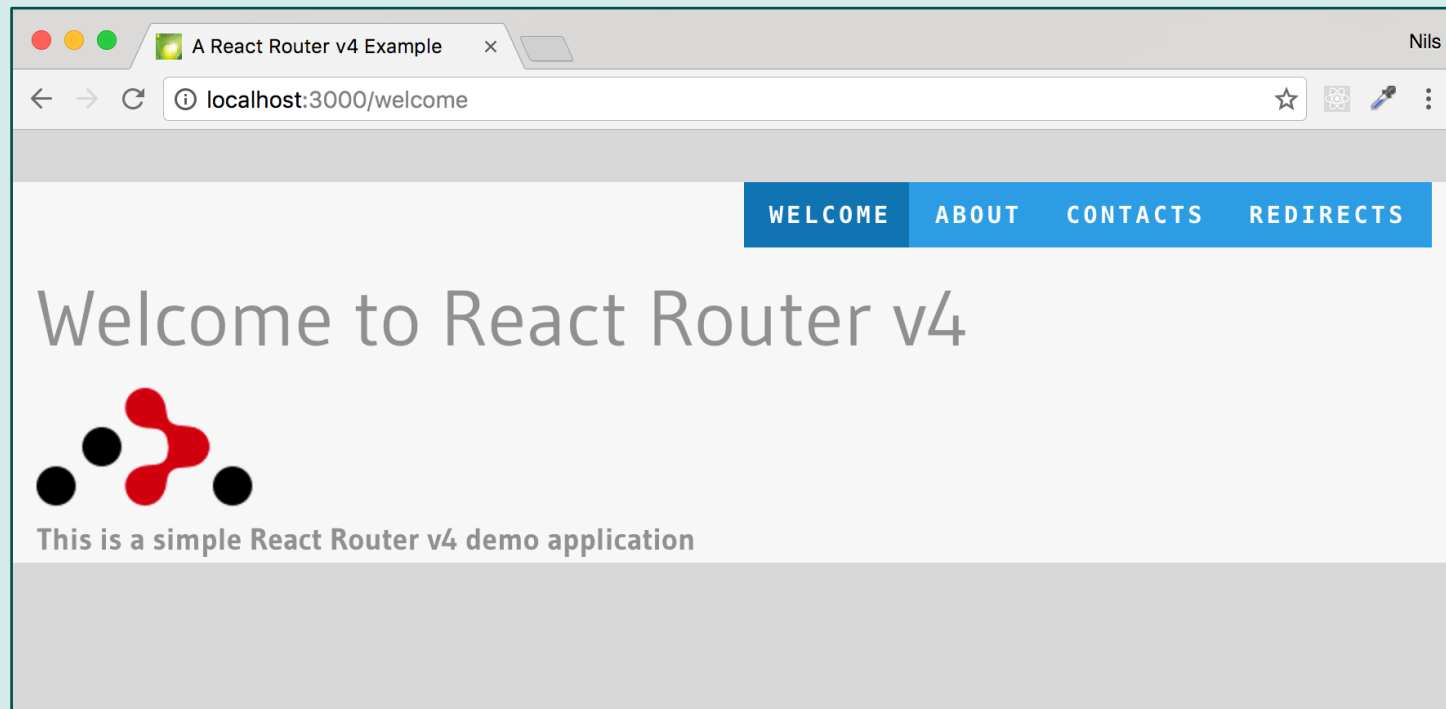


@NILSHARTMANN

REACT ROUTER v4

<http://bit.ly/react-rr4>



[HTTPS://GITHUB.COM/NILSHARTMANN/REACT-RR4-EXAMPLE](https://github.com/nilshartmann/react-rr4-example)

RECAP: REACT ROUTER V2 / V3

Explicit history object

```
import  
  {Router, Route, browserHistory, ...} from 'react-router';
```

RECAP: REACT ROUTER V2 / V3

Global (central) route config

```
import
  {Router, Route, browserHistory, ...} from 'react-router';

const routes = <Router history={browserHistory}>
  <Route component={App}>

    </Route>
  </Router>;
```

RECAP: REACT ROUTER V2 / V3

Nested routes

```
import
  {Router, Route, browserHistory, ...} from 'react-router';

const routes = <Router history={browserHistory}>
  <Route component={App}>
    <Route path='/about' component={AboutPage}>
      <Route path='more' component={More} />
      <Route path='why' component={Why} />
    </Route>
    <Route path='*' component={NotFound} />
  </Route>
</Router>;
```

Links

```
import { Link } from 'react-router';

const AboutPage = () => (
  <div>
    <Link to='/about/why'>Why</Link>
    <Link to='/about/source'>Source</Link>
    <Link to='/about/more'>More</Link>
  </div>
);
```

```
npm install --save react-router@next
```

REACT ROUTER  v4

REACT ROUTER V4

Current state: Alpha Version

- Version 4.0.0-**alpha6** (Released: yesterday...)
- Be careful! Use at your own risk!

Docs and sources

- <https://react-router.now.sh/>
- <https://github.com/ReactTraining/react-router/tree/v4>

REACT ROUTER V4: ROUTER

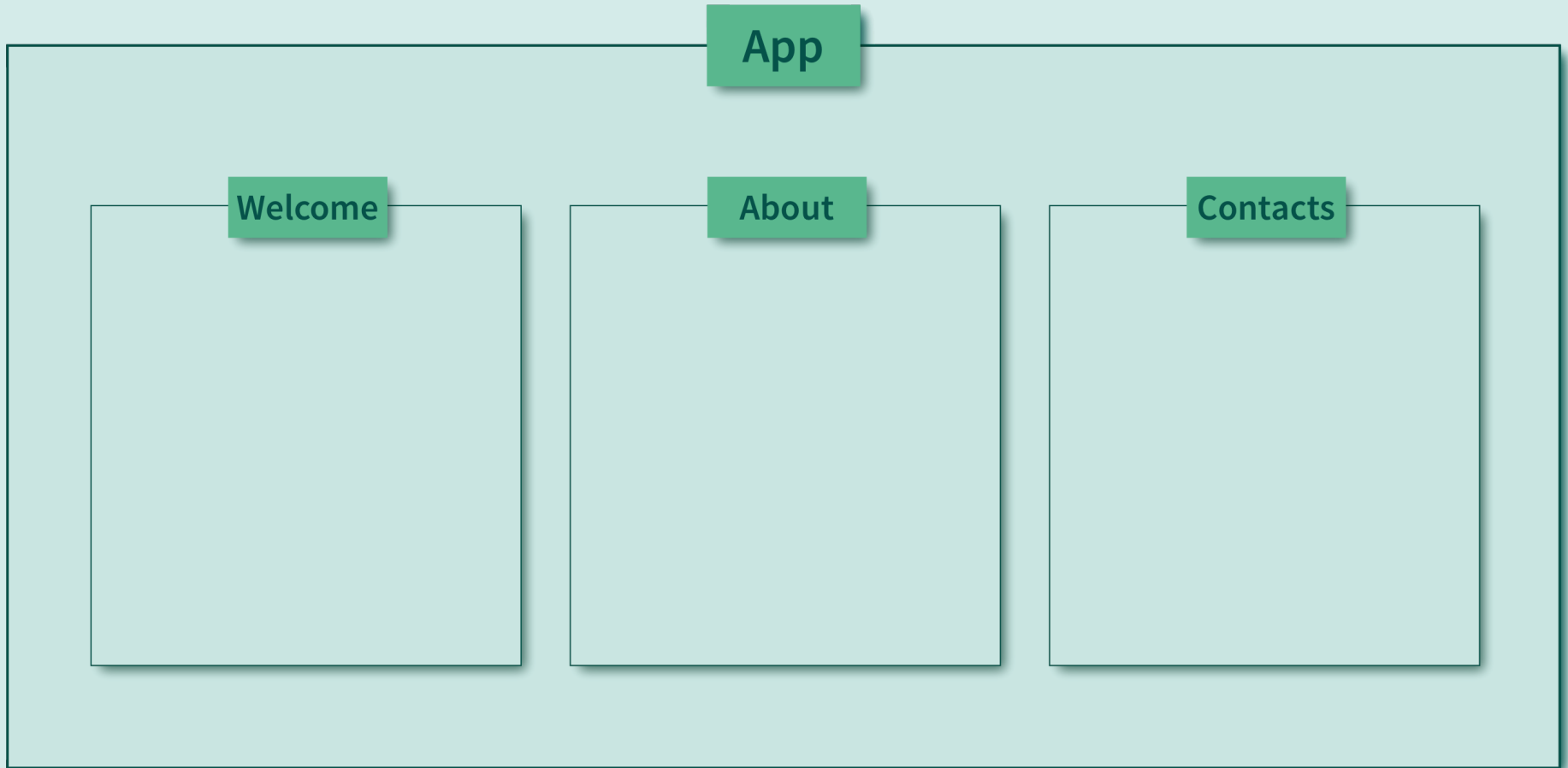
```
import BrowserRouter from 'react-router/BrowserRouter';
import App from './App';

ReactDOM.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>,
  document.getElementById('... ')
);
```

Router as new top-level component

- No centralized route config anymore!
- Instead of History use appropriate Router component:
 - BrowserRouter, HashRouter, MemoryRouter, ServerRouter (new!)

MATCHING



MATCHING 1

App.jsx

```
// 'regular' component
const App = () => (
  <main>
    <header>. . .</header>
    // add components here according to current route
    ...

  </main>
);
```

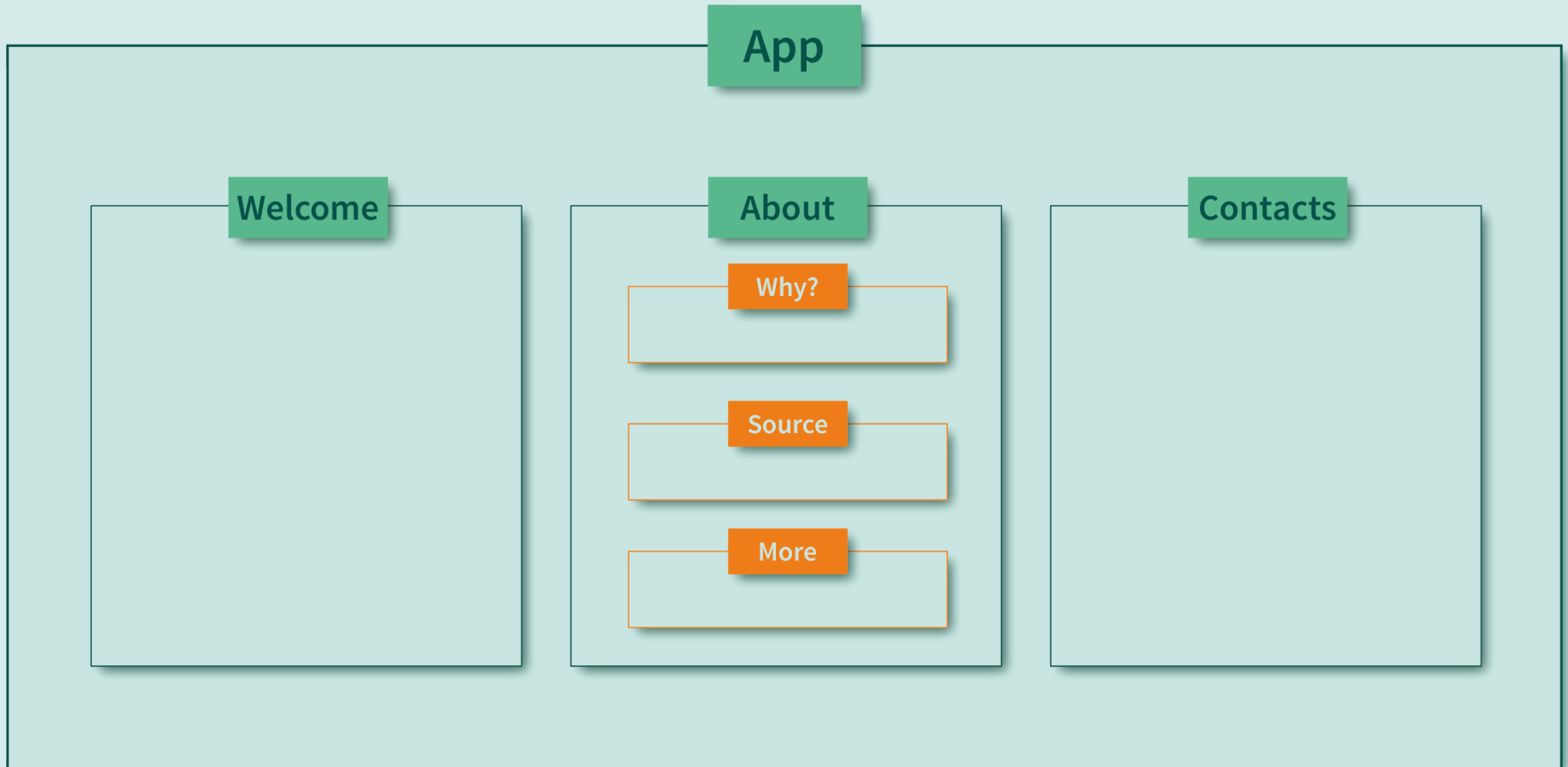
MATCHING - 2

App.jsx

```
import Match from 'react-router/Match';

// 'regular' component
const App = () => (
  <main>
    <header>. . .</header>
    // add components here according to current route
    <Match pattern='/welcome' component={ Welcome } />
    <Match pattern='/about' component={ About } />
    <Match pattern='/secret' component={ Secret } />
  </main>
);
```

MATCHING - "SUB ROUTES"



MATCHING - "SUB ROUTES"

About.jsx

```
import Match from 'react-router/Match';

// 'regular' component
const About = ({pathname}) => (
  <div>
    <h1>About</h1>
    <div>
      <Match pattern={`${pathname}/why`} component={Why} />
      <Match pattern={`${pathname}/more`} component={More} />
    </div>
  </div>
);
```

MATCHING - MULTIPLE MATCHES

Page.jsx

```
import Match from 'react-router/Match';

const Layout = () => (
  <div>
    <aside>
      <Match pattern='/about' component={ContactBlock} />
      <Match pattern='/source' component={GitHubLogo} />
    </aside>
    <main>
      <Match pattern='/about' component={AboutPage} />
      <Match pattern='/source' component={SourcePage} />
    </main>
  </div>
);
```

MATCHING - EXAMPLES

Matches for /:

```
<Match pattern='/' exactly component={ ... } />
```

Matches for /about, **NOT** for /about/:

```
<Match pattern='/about' exactly component={ ... } />
```

Matches for /about, /about/, /about/me, /about/react/router/v4:

```
<Match pattern='/about' component={ ... } />
```

Dynamic Segments

Matches for /contacts/1, /contacts, contacts/1/a:

```
<Match pattern='/contacts/:id?' component={ ... } />
```

Matches for /contacts/1, /contacts, **NOT** for /contacts/:

```
<Match pattern='/contacts/:id?' exactly component={ ... } />
```


MATCHING - PROPERTIES

<Match /> passes properties to rendered component, e.g:

- `pathname`: The path that has been matched ('/about/why')
- `pattern`: The pattern that has been matched ('why')
- `location`: The location object (from History project)
- `params`: Values from the dynamic segments in pattern ('.../:id')

MATCHING – RENDER FUNCTION

`<Match render = { (props) => { ... } />:`

- Specifies a **function** rather than a **component**
- Function receives same parameters as component
- Returned component is rendered

MATCHING – RENDER FUNCTION

Example 1: Conditional rendering

```
<Match pattern='/secret' render={ () => {  
  if (!loggedIn) {  
    return <LoginForm />;  
  }  
  
  return <SecretPage />;  
} } />
```

MATCHING-RENDER FUNCTION

Example 2: Pass properties to component

```
const Page = ({allContacts}) => (  
  <div>  
    <header>...</header>  
    <main>  
      <Match pattern='/contacts' render={ props => (  
        <ContactsPage contacts={allContacts}  
          currentContactId={props.params.id} />  
      ) } />  
    </main>  
  </div>  
) ;
```

REDIRECT

`<Redirect to='/some/path' />`

- Redirect to a specified path (or location)
- Works like a "regular component"

Example 1: Use in `<Match />`-function

```
import Redirect from 'react-router/Redirect';
```

```
const App = () => (  
  <main>  
    <header>. . .</header>  
    <Match pattern='/' render={() => <Redirect to='/index' /> } />  
    <Match pattern='/index' component={ IndexPage } />  
  </main>  
>);
```

REDIRECT

`<Redirect to='/some/path' />`

- Redirect to a specified path (or location)
- Works like a "regular component"

Example 2: Use in "regular" render function

```
import Redirect from 'react-router/Redirect';
```

```
class Login extends React.Component {  
  render() {  
    const { loginSuccessful } = this.state;  
    const { redirectAfterLogin } = this.props;  
    return loginSuccessful ?  
      <Redirect to={redirectAfterLogin} /> : <LoginForm />;  
  }  
}
```

REDIRECT - IMPERATIVE

Alternative: Redirect via API call

- Access to router via React Context (key: **router**)
- Specifies two functions: **transitionTo** and **replaceWith**

Example

```
import { routerContext } from 'react-router/PropTypes';

class Login extends React.Component {
  static contextTypes = { router: routerContext };

  doRedirectAfterLogin() {
    this.context.router.transitionTo('/...');
  }

  render() { . . . }
}
```

MATCHING – NO HIT: MISS

`<Miss component={ ... } />`

`<Miss render={ (params) => { ... } } />`

- Is rendered when no `<Match />` hits
- Takes either a component or a function (as in `<Match />`)
- Receives a **location** object with unmatched path

<http://localhost:3000/not-there?name=klaus>

location:

hash: ""

key: undefined

pathname: "/not-there"

query:

name: "klaus"

search: "?name=klaus"

state: null

MATCHING – NO HIT: MISS

Example: <Miss />

```
import Miss from 'react-router/Miss';

const NoMatch =
  ({location}) => <b>Not found: {location.pathname}</b>;

const App = () => (
  <main>
    <Match pattern='/index' component={ IndexPage } />
    <Match pattern='/welcome' component={ Welcome } />
    <Miss component={ NoMatch } />
  </main>
);
```

LINKS

`<Link to='...' />`:

- Renders a link (`...`) to specified path
- Properties almost identical to React Router v3
 - `to`, `activeClassName`, `activeStyle`

```
import Link from 'react-router/Link';
```

```
const About = () => (
```

```
  <span>
```

```
    Click here <Link
```

```
      to='/about/more'
```

```
      activeClassName='active'
```

```
      activeStyle={{fontWeight: 'bold'}}>for info</Link>
```

```
  </span>
```

```
);
```

LINKS - ACTIVE LINK

```
<Link to='/' isActive={(params) => {...}} />
```

- Callback function to determine if Link should be 'active'
 - **location, to:** Current and target (as **location**-object)
 - **props:** All properties passed to Link component

```
import Link from 'react-router/Link';
```

```
<Link to='/contact' data-alias='/address'  
  isActive={ (location, to, props) => (  
    location.pathname === to.pathname  
    || location.pathname === props['data-alias']  
  )}  
>
```

LINKS: CUSTOM COMPONENT

`<Link>{(params) => { . . . } }</Link>`: Render custom component

- Function as (the only) child to `<Link />`
- Render and return an own component instead of `<a>...`
- Useful for menus, buttons etc
- Receives parameters needed to build the component
 - `isActive`, `location`, `href`, `onClick` ...

LINKS: CUSTOM COMPONENT

Example: A Menu Entry

```
const MenuEntry = ({ label, to }) => (  
  <Link to={to}>{params => (  
    <li className={params.isActive ? 'menu active' : 'menu'}>  
      <a href={params.href} onClick={params.onClick}>{label}</a>  
    </li>  
  )}  
  </Link>  
);
```

```
<MenuEntry to='/contacts/1'  
  label='Klaus Peterson' />  
<MenuEntry to='/contacts/2'  
  label='Ursula Meier' />
```

...

Contacts

Klaus Peterson

Ursula Meier

Michel Svenson

Pierre Michel Lassoga

RELATIVE PATHS

Relative paths should be avoided

- "tricky to handle"
- Not (fully) implemented currently

Workaround: use 'pathname'

```
const About = ( { pathname } ) => (  
  <header>  
    <Link to={` ${pathname} /more`} >More...</Link>  
  <header>  
    <main>  
      <Match pattern={` ${pathname} /more`} component={More} />  
    </main>  
  );
```

SUMMARY

Still alpha!

- Expect to find bugs
- Not that much documentation (but looks promising)

Good improvements over v2/v3

- Decentralized matching
- Custom link components
- Matching with functions rather than components

Could be better?!

- No 'router' as component properties
- Custom link components
- Relative links and matching

Thank you and happy routing!

<http://bit.ly/react-rr4>

Questions?

@NILSHARTMANN