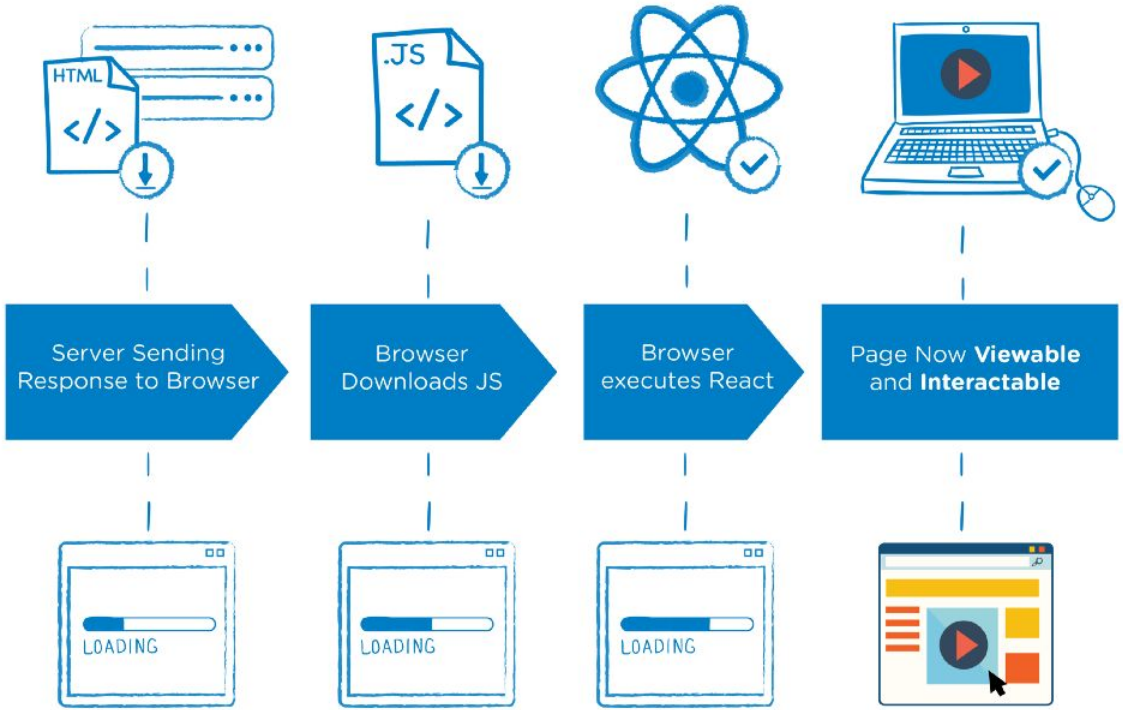
React server side rendering

Koa, webpack, routing, redux

create-react-app

CSR


HTML
.JS
Server Sending Response to Browser
Browser Downloads JS
Browser executes React
Page Now **Viewable** and **Interactable**
LOADING
LOADING
LOADING

Benefits CSR

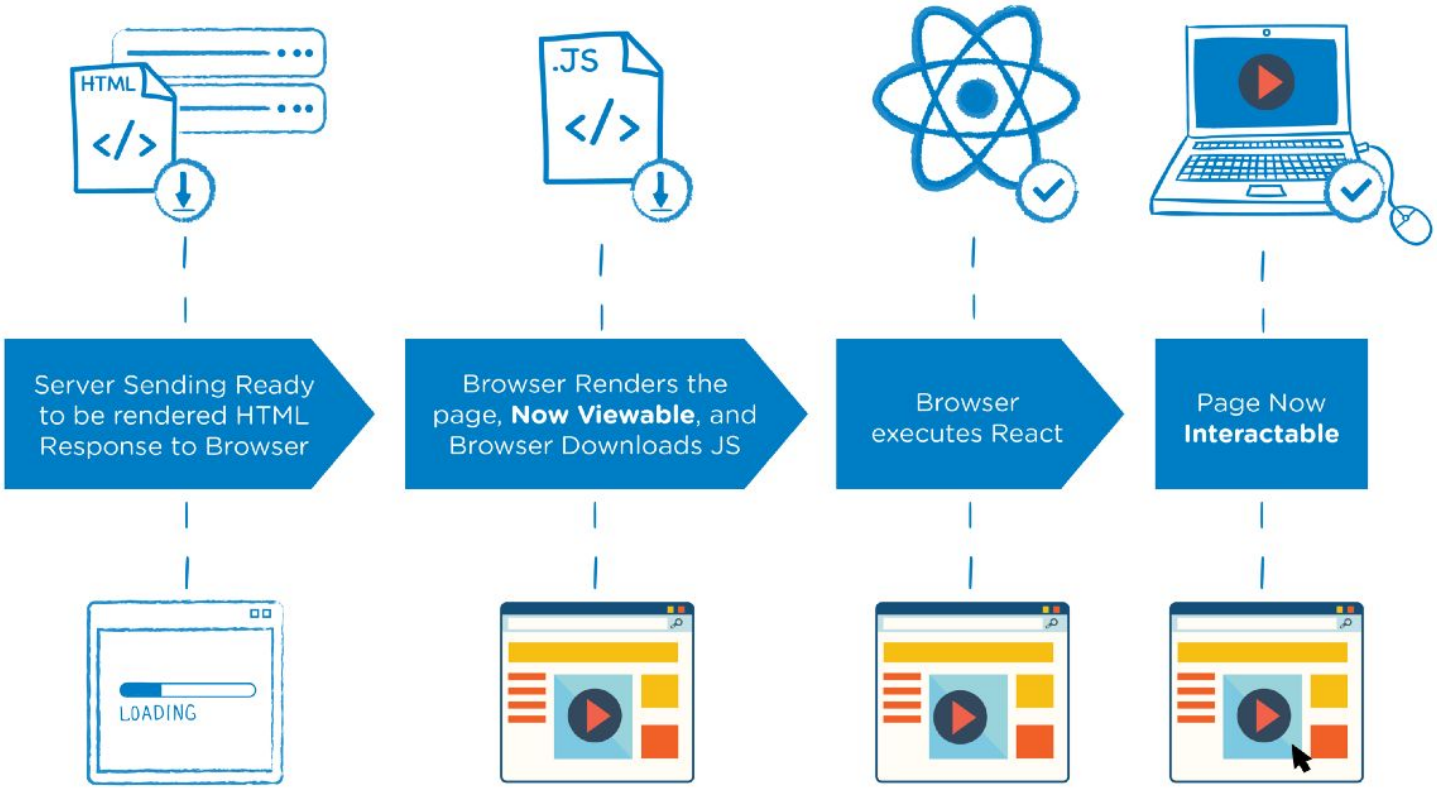
- Decreased load on the server
- Good UI/UX interactivity
- Static hosting

Disadvantages CSR

- Slow initial render
- Non-SEO friendly

SSR + SPA = Universal App

SSR


HTML
.JS
Server Sending Ready to be rendered HTML Response to Browser
Browser Renders the page, **Now Viewable**, and Browser Downloads JS
Browser executes React
Page Now **Interactable**
LOADING

Benefits SSR

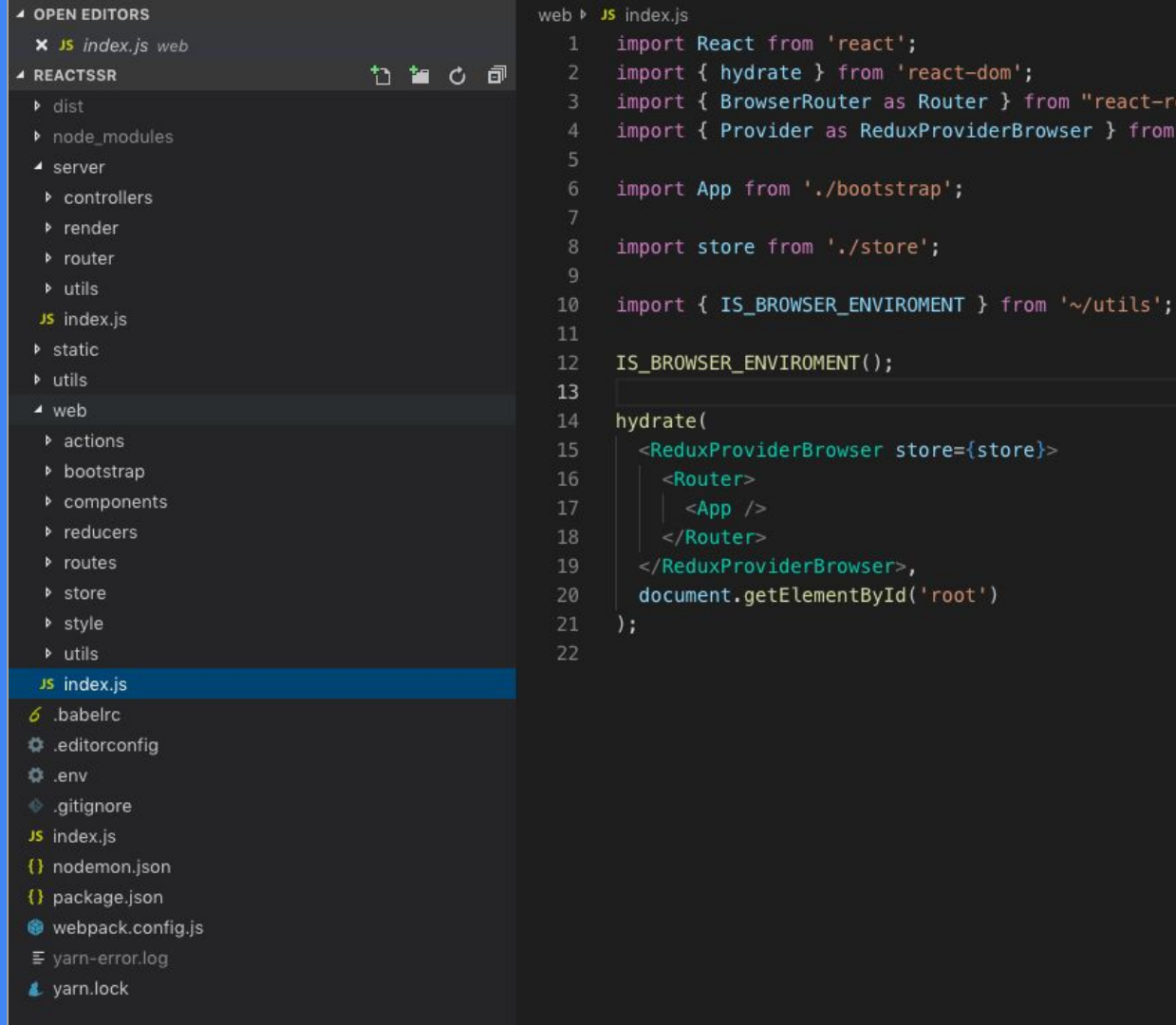
- Fast first paint
- Better perceived perfomance
- SEO friendly

Disadvantages SSR

- Increased load on the server
- Latency due to page request / reloads
- Non interactive page until React excutes

React/Redux SSR app via Koa.js

Project skeleton



Basic setup

Wepback

~/webpack.config.js

```
module.exports = {
  entry: './web/index.js',
  output: {
    path: path.resolve(__dirname, buildFolder),
    publicPath: '/'
  },
  plugins: [
    new CleanWebpackPlugin(),
    new CopyWebpackPlugin([
      { from: './static', to: './' }
    ]),
    new Dotenv()
  ],
  mode: process.env.NODE_ENV,
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /node_modules/,
        loader: "babel-loader"
      },
      {
        test: /\.scss$/,
        use: [
          {
            loader: "style-loader"
          },
          {
            loader: "css-loader"
          },
          {
            loader: "sass-loader"
          }
        ]
      }
    ]
  }
}
```

Babel

~/.babelrc

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "targets":
        {
          "node": "current"
        }
      }
    ],
    "@babel/preset-react"
  ],
  "plugins": [
    ["babel-plugin-root-import"],
    ["@babel/plugin-proposal-decorators", { "legacy": true }],
    ["@babel/plugin-proposal-class-properties", {  "loose": true }]
  ]
}
```

~/index.js

```
require("@babel/register")();
require('dotenv').config();
require('./server');
```

Client

~/web/bootstrap/index.js

~/web/index.js

web ▸ bootstrap ▸ JS index.js ▸ ...

```
1  import React from 'react';
2
3  const App = () => <h1>Awsome app</h1>;
4
5  export default App;
6
```

web ▸ JS index.js

```
1  import React from 'react';
2  import { hydrate } from 'react-dom';
3
4  import App from './bootstrap';
5
6  hydrate(
7    <App />,
8    document.getElementById('root')
9  );
10
```

Server (bootstrap)

~/server/index.js

server ▸ JS index.js ▸ app.listen() callback

```
1  import Koa from 'koa';
2  import serve from 'koa-static';
3
4  import buildWeb from './utils/buildWeb';
5  import renderMiddleware from './render';
6
7  const app = new Koa();
8
9  app.use(serve('dist'));
10 app.use(renderMiddleware);
11
12 const port = process.env.APP_PORT || 3000;
13 app.listen(port, async error => {
14   if (!error) {
15     await buildWeb();
16
17     console.log(`The server has been started at port - ${port}`);
18   } else {
19     console.log(error);
20   }
21 });
22
```

buildWeb.js

~/server/utils/buildWeb.js

```
1  import webpack from 'webpack';
2  import config from '~/webpack.config.js';
3
4  const buildWeb = () => new Promise((resolve, reject) => {
5    console.log('starting to build web...');
6
7    webpack(config, (err, stats) => {
8      const error = err || stats.hasErrors();
9      if (error) {
10       reject(error);
11     }
12     console.log('Web is built!');
13     resolve();
14   });
15 });
16
17 export default buildWeb;
18
```

Rendering on server

~/server/render/render.js

```
import React from "react";
import { renderToString } from 'react-dom/server';

import App from '~/web/bootstrap';
import template from './template';

const renderPage = async () => {
  const html = renderToString(<App />);

  const rendered = template({ html });

  return rendered;
}

const renderMiddlware = async (ctx) => {
  if (ctx.response.status === 404) {
    const renderedPage = await renderPage(ctx.request.url);

    ctx.response.status = 200;
    ctx.type = 'html';
    ctx.body = renderedPage;
  }
}

export default renderMiddlware;
```

HTML template

~/server/render/template.js

```
export default ({ html }) => `
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8">
      <title>React Koa SSR</title>
    </head>

    <body>
      <div id="root">${html}</div>
      <script src="${process.env.ROOT_URL}main.js"></script>
    </body>
  </html>
`
```

Styles (SASS)

Importing via webpack

~/web/index.js

```
import React from 'react';
import { hydrate } from 'react-dom';

import App from './bootstrap';

import '~/web/style/style.scss'; // Is that a good idea? :(

hydrate(
  <App />,
  document.getElementById('root')
);
```

StyleHandler

~/server/utils/StyleHandler.js

```
import sass from 'node-sass';

class StyleHandler {
  static removeCommentsInStyles = str =>
    str.replace(/\/\*[^*]*\*+([^\/][^*]*\*+)*\//, '');

  _styleData = '';

  getStyleData = () => this._styleData;

  loadStyles = () => new Promise((resolve, reject) => {
    console.log('Starting to compile Sass...')
    sass.render({
      file: 'web/style/index.scss',
      outputStyle: 'compressed'
    }, (error, result) => {
      if (error) {
        reject(error);
      } else {
        const cssData = result.css.toString();
        const css = StyleHandler.removeCommentsInStyles(cssData);
        this._styleData = css;

        console.log('Sass is compiled!');
        resolve(css);
      }
    })
  })
}

export default new StyleHandler();
```

StyleHandler (usage)

~/server/index.js

~/server/router/index.js

```
const port = process.env.APP_PORT || 3000;
app.listen(port, async error => {
  if (!error) {
    await StyleHandler.loadStyles();
    await buildWeb();

    console.log(`The server has been started at port - ${port}`);
  } else {
    console.log(error);
  }
});
```

```
import koaRouter from 'koa-router';
import StyleHandler from '~/server/utils/StyleHandler';

const router = new koaRouter();

router.get('/style.css', (ctx) => {
  const styleData = StyleHandler.getStyleData();

  ctx.response.status = 200;
  ctx.type = 'text/css';
  ctx.body = styleData;
});
```

Update template

~/server/render/template.js

```
export default ({ html }) => `
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8">
      <title>React Koa SSR</title>
      <link rel="stylesheet" href="${process.env.ROOT_URL}style.css">
    </head>

    <body>
      <div id="root">${html}</div>
      <script src="${process.env.ROOT_URL}main.js"></script>
    </body>
  </html>
`
```

Other solutions

- Styled components (need to update babel config)
- Isomorphic-style-loader (need to build server from webpack)

React Router

Make some routes

~/web/bootstrap/index.js

```
import React from 'react';
import { Switch, Route } from "react-router-dom";

const App = () => (
  <Switch>
    <Route
      component={AwesomeComponent}
      path='/somewhere'
      exact
    />
  </Switch>
);

export default App;
```

Setup router (client)

~/web/index.js

```
import React from 'react';
import { hydrate } from 'react-dom';
import { BrowserRouter as Router } from "react-router-dom";

import App from './bootstrap';

hydrate(
  <Router>
    <App />
  </Router>,
  document.getElementById('root')
);
```

Setup router (server)

~/server/render.js

```
import { StaticRouter } from 'react-router-dom';

import template from './template';

import App from '~/web/bootstrap';

const renderPage = async (url) => {
  const html = renderToString(
    <StaticRouter context={{}} location={url} >
      <App />
    </StaticRouter>
  );

  const rendered = template({
    html
  });

  return rendered;
}

const renderMiddlware = async (ctx) => {
  if (ctx.response.status === 404) {
    const renderedPage = await renderPage(ctx.request.url);

    ctx.response.status = 200;
    ctx.type = 'html';
    ctx.body = renderedPage;
  }
}

export default renderMiddlware;
```

Redux

Same scripts on both sides

~/utils/index.js

```
const isServer = typeof window === 'undefined';

export const switcher = ({ browser, server }, ...args) => {
  return isServer ?
    (server && server(...args)) :
    (browser && browser(...args))
};

export const onlyBrowser = (target, key, descriptor) =>  {
  const originalMethod = descriptor.value;

  descriptor.value = function(...args) {
    return switcher({
      browser: () => originalMethod.apply(this, args)
    });
  }

  return descriptor;
};

export const IS_BROWSER_ENVIROMENT = () => {
  if (isServer) {
    throw new Error('That file should be used only in the browser side')
  }
};
```

IS_BROWSER_ENVIRONMENT

~/web/index.js

```
import React from 'react';
import { hydrate } from 'react-dom';
import { BrowserRouter as Router } from "react-router-dom";

import App from './bootstrap';

import { IS_BROWSER_ENVIROMENT } from '~/utils';

IS_BROWSER_ENVIROMENT();

hydrate(
  <Router>
    <App />
  </Router>,
  document.getElementById('root')
);
```

Create fake controller

~/server/controllers/post.js

```
const posts = [
  {
    id: 1,
    title: 'Some title',
    body: 'some awesome text'
  },
  {
    id: 2,
    title: 'Something',
    body: 'some awesafdsdf'
  },
  {
    id: 3,
    title: 'Some fwadtitle',
    body: 'some 2342awesome text'
  },
  {
    id: 4,
    title: 'Some asdftitle',
    body: 'some a asdfwesome text'
  }
]

export const getPosts = () => new Promise(resolve =>
  setTimeout(() => resolve(posts), 100)
);
```

Create posts route

~/server/router/index.js

```
import koaRouter from 'koa-router';
import StyleHandler from '~/server/utils/StyleHandler';
import { getPosts } from '~/server/controllers/post';

const router = new koaRouter();

router.get('/posts', async (ctx) => {
  const posts = await getPosts();

  ctx.response.status = 200;
  ctx.body = posts;
});

router.get('/style.css', (ctx) => {
  const styleData = StyleHandler.getStyleData();

  ctx.response.status = 200;
  ctx.type = 'text/css';
  ctx.body = styleData;
});

export default router;
```

Creating store

~/web/store/createStore.js

~/web/store/store.js

```
import { createStore, applyMiddleware } from 'redux';
import thunkMiddleware from 'redux-thunk';

import rootReducer from '~/web/reducers';

export default (initialState) =>
  createStore(
    rootReducer,
    initialState,
    applyMiddleware(thunkMiddleware)
  );
```

```
import createStore from './createStore';
import { IS_BROWSER_ENVIRONMENT } from '~/utils';

// only in browser
IS_BROWSER_ENVIRONMENT();

const store = createStore();

export default store;
```

Add some reducers, actions and action creators

~/web/actions/home/index.js

```
import axios from 'axios';
import * as types from './types';

export const setPosts = posts => ({
  type: types.SET_POSTS,
  payload: posts
});

export const fetchPosts = () => async (dispatch) => {
  try {
    const response = await axios.get(`${process.env.ROOT_URL}/posts`);

    const posts = response.data;

    dispatch(setPosts(posts));
  } catch (e) {

  }
};
```

Setup Redux (client)

~/web/index.js

```
import React from 'react';
import { hydrate } from 'react-dom';
import { BrowserRouter as Router } from "react-router-dom";
import { Provider as ReduxProviderBrowser } from 'react-redux';

import App from './bootstrap';

import store from './store';

import { IS_BROWSER_ENVIRONMENT } from '~/utils';

IS_BROWSER_ENVIRONMENT();

hydrate(
  <ReduxProviderBrowser store={store}>
    <Router>
      <App />
    </Router>
  </ReduxProviderBrowser>,
  document.getElementById('root')
);
```

Setup Redux (server)

~/server/render/render.js

```
import { Provider as ReduxProviderServer } from 'react-redux';

import generateHead from '~/server/utils/generateHead';
import template from './template';

import App from '~/web/bootstrap';
import createStore from '~/web/store/createStore';

import webRoutes from '~/web/routes';

const renderPage = async (url) => {
  const store = createStore();

  const html = renderToString(
    <ReduxProviderServer store={store}>
      <StaticRouter context={{}} location={url} >
        <App />
      </StaticRouter>
    </ReduxProviderServer>
  );

  const rendered = template({
    html,
    state: store.getState()
  });

  return rendered;
}
```

Update HTML template

~/server/render/template.js

```
export default ({ html, state }) => `
  <!DOCTYPE html>
  <html>
    <head>
      <meta charset="utf-8">
      <title>React Koa SSR</title>
      <link rel="stylesheet" href="${process.env.ROOT_URL}/style.css">
    </head>

    <body>
      <div id="root">${html}</div>
      <script id="redux-data-script">
        window.__REDUX_DATA__ = ${JSON.stringify(state)}
      </script>
      <script src="${process.env.ROOT_URL}/main.js"></script>
    </body>
  </html>
`
```

Set redux data from the server on the client

~/web/store/store.js

```
import createStore from './createStore';
import { IS_BROWSER_ENVIRONMENT } from '~/utils';

// only in browser
IS_BROWSER_ENVIRONMENT();

const deleteInitialReduxData = () => {
  delete window.__REDUX_DATA__;

  const element = document.getElementById('redux-data-script');
  element.parentNode.removeChild(element);
}

const store = createStore(window.__REDUX_DATA__);

deleteInitialReduxData();

export default store;
```

Getting/fetching data

Make web routes array

~/web/routes/routes.js

~/web/bootstrap/index.js

```
import Home from '~/web/components/Home';
import About from '~/web/components/About';

export default [
  {
    component: Home,
    path: '/',
    exact: true
  },
  {
    component: About,
    path: '/about',
    exact: true
  },
  {
    component: () => (
      <div className="text-center">
        <h1>404</h1>
        <h2>Page not found</h2>
      </div>
    ),
  }
];
```

```
import routes from '~/web/routes';

const App = () => (
  <Switch>
    {routes.map(({ path, ...data }) => (
      <Route
        key={path + 'route'}
        path={path}
        {...data}
      />
    ))}
  </Switch>
);

export default App;
```

Add “initialLoad” static method and “withInitialLoading” decorator to the route component

~/web/components/Home/Home.js

```
import { withInitialLoading } from '~/web/utils';

import { fetchPosts } from '~/web/actions/home';

@withInitialLoading
@connect(state => ({ data: state.home.data }))
class Home extends Component {

  static initialLoad = ({ store, match }) => {
    console.log(match); // route params
    return store.dispatch(fetchPosts());
  };

  static defaultProps = {
    data: []
  }

  render() {
    const { data } = this.props;

    return (
      <div className="m-auto w-75">
        <h1
          className="text-center"
        >
          It is a home page
        </h1>
        <div>Some awesome home page with some awesome description like never before you havent seen
        <Link to="/about">About</Link>
        <div className="list">
          {data.map(({ id, title, body }) => (
            <div key={id} className="item">
              <div className="title text-center mb-2">{title}</div>
              <div className="text">{body}</div>
            </div>
          ))}
```

withInitialLoading

~/web/utils/withInitialLoading.js

```
import { withRouter } from 'react-router-dom';

import { onlyBrowser } from '~/utils';

export const withInitialLoading = IncommingComponent => {
  const { initialLoad } = IncommingComponent;
  if (!initialLoad) {
    return IncommingComponent;
  }

  @withRouter
  class Wrapper extends Component {
    static initialLoad = initialLoad;

    componentDidMount() {
      this.load();
    }

    @onlyBrowser load = () => {
      const { match } = this.props;
      const { default: store } = require('~/web/store');

      initialLoad({
        store,
        match
      });
    }

    render() {
      return (
        <IncommingComponent load={this.load} {...this.props} />
      );
    }
  }

  return Wrapper;
}
```

Calling initialLoad from the server

~/server/render/render.js

```
import webRoutes from '~/web/routes';
import { StaticRouter, matchPath } from 'react-router-dom';

const renderPage = async (url) => {
  const store = createStore();

  const matchedRoute = webRoutes.find(route => route.path && matchPath(url, route)) || {};
  const { component } = matchedRoute;

  if (component) {
    const { initialLoad } = component;

    if (initialLoad) {
      const match = matchPath(url, matchedRoute);
      await initialLoad({
        store,
        match
      });
    }
  }

  const html = renderToString(
    <ReduxProviderServer store={store}>
      <StaticRouter context={{}} location={url} >
        <App />
      </StaticRouter>
    </ReduxProviderServer>
  );

  const rendered = template({
    html,
    state: store.getState(),
```

Update fetch posts method for getting date in better way

~/web/action/home/index.js

```
export const fetchPosts = () => async (dispatch) => {
  try {
    const posts = await switcher({
      browser: async () => {
        const response = await axios.get(`${process.env.ROOT_URL}/posts`);

        return response.data
      },
      server: () => require('~/server/controllers/post').getPosts()
    });


    dispatch(setPosts(posts));
  } catch (e) {

  }
};
```

SEO

react-helmet

Nested or latter components will override duplicate changes:

```
<Parent>
    <Helmet>
        <title>My Title</title>
        <meta name="description" content="Helmet application" />
    </Helmet>

    <Child>
        <Helmet>
            <title>Nested Title</title>
            <meta name="description" content="Nested component" />
        </Helmet>
    </Child>
</Parent>
```

outputs:

```
<head>
    <title>Nested Title</title>
    <meta name="description" content="Nested component">
</head>
```

Add helmet to components

~/web/components/Home/Home.js

```
  render() {
    const { data } = this.props;

    return (
      <div className="m-auto w-75">
        <Helmet>
          <title>Awesome page</title>
        </Helmet>
        <h1
          className="text-center"
        >
          It is a home page
        </h1>
        <div>Some awesome home page with some awesome description like never before y
        <Link to="/about">About</Link>
        <div className="list">
          {data.map(({ id, title, body }) => (
            <div key={id} className="item">
              <div className="title text-center mb-2">{title}</div>
              <div className="text">{body}</div>
            </div>
          ))}
        </div>
      </div>
    )
  }
}


export default Home;
```

Helmet (server)

~/server/utils/generateHead.js

```
import { Helmet } from 'react-helmet';

const generateHead = () => {
  const headData = Helmet.renderStatic();
  const head = Object.keys(headData).reduce((result, key) => {
    const el = headData[key].toString();

    return el ? result + el : result;
  }, '');

  return head;
};

export default generateHead;
```

~/server/render/render.js

```
const html = renderToString(
  <ReduxProviderServer store={store}>
    <StaticRouter context={{}} location={url} >
      <App />
    </StaticRouter>
  </ReduxProviderServer>
);

const rendered = template({
  html,
  state: store.getState(),
  head: generateHead()
});
```

Update HTML template

~/server/render/template.js

```
export default ({ html, state, head }) => `
  <!DOCTYPE html>
  <html>
    <head>
        <meta charset="utf-8">
        <title>React Koa SSR</title>
        <link rel="stylesheet" href="${process.env.ROOT_URL}/style.css">
        ${head}
    </head>

    <body>
        <div id="root">${html}</div>
        <script id="redux-data-script">
          window.__REDUX_DATA__ = ${JSON.stringify(state)}
        </script>
        <script src="${process.env.ROOT_URL}/main.js"></script>
    </body>
  </html>
`
```

What is the next step?

Hm...

Thanks!

https://github.com/yurkagon/React-Redux_Koa_SSR_boilerplate