CSR with ReactJS - a case study

Deepak Parasam

A bit about myself

- Software engineering lead at SparkPost
- Interested in web development and distributed computing
- Previously worked at IBM, Amazon
- Fascinated with ReactJS!

@deepak_pn

https://thinkerbits.com/

Overview

- CSR vs. SSR - quick recap
- Case Study - PowerMTA web monitor
    - Architecture
    - Design goals
    - Building the app
    - Limitations
    - Demo
- Q&A

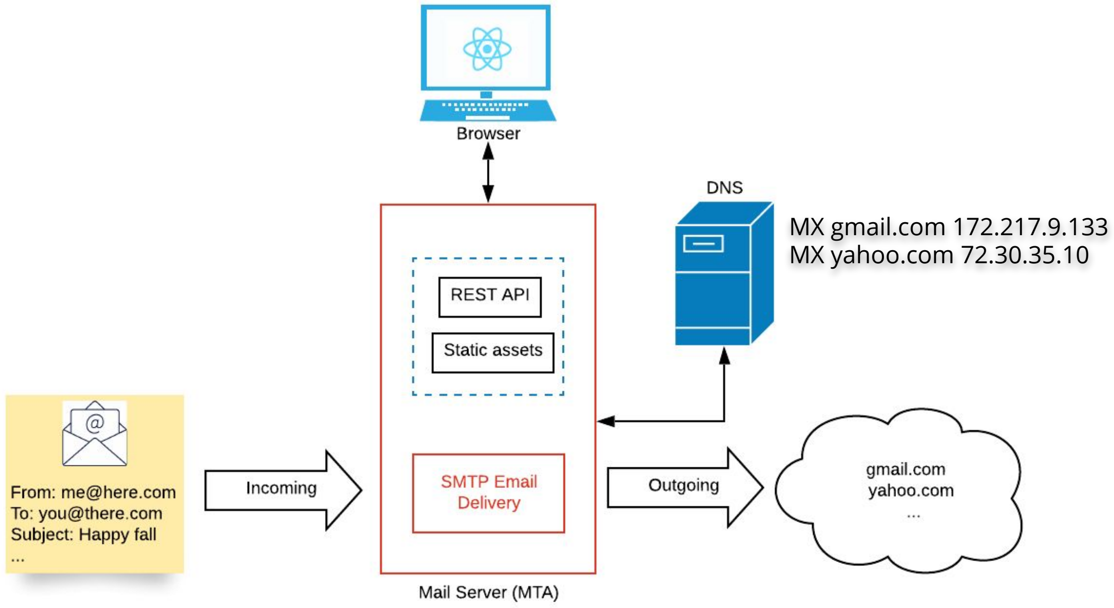
CSR vs. SSR

SSR: Browser gets and renders the HTML while JS is downloaded and executed

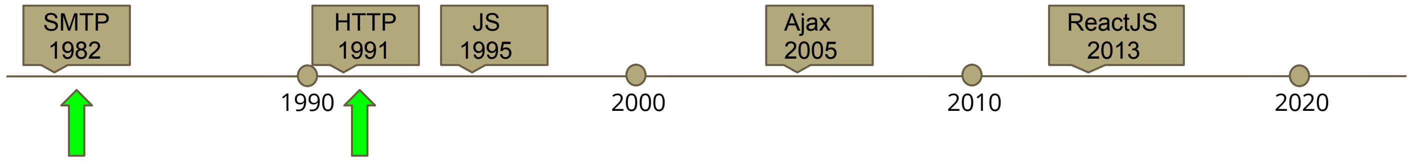
CSR: Browser gets a pretty empty document with links to your JS

https://medium.com/walmartlabs/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8

PowerMTA Web Monitor Architecture



Case Study - PowerMTA Web Monitor



Design goals

- Modernize UI: rich UX for customers, and improve developer productivity
- Deployments: easy/fast
    - Runtime dependencies
    - Matters more with large # of servers
- Minimal resource footprint
    - should not impact email delivery

Concerns with CSR not significant - used in a LAN setting

Building the app

- Create-React-App
  - Serve static assets (package.json: “homepage”: “/ui”)
    - /ui => build/index.html
    - /ui/static/* => build/static/*
  - Serve dynamic content through the REST API
  - HTTP endpoint
    - HTTP Headers: Content-Type, Content-Encoding
    - Full control over TLS support
    - https://developer.ibm.com/blogs/openssl-111-has-landed-in-nodejs-master-and-why-its-important-for-nodejs-lts-releases/
- Routing with BrowserRouter
  - <Switch>
  - Unknown URL path => return build/index.html

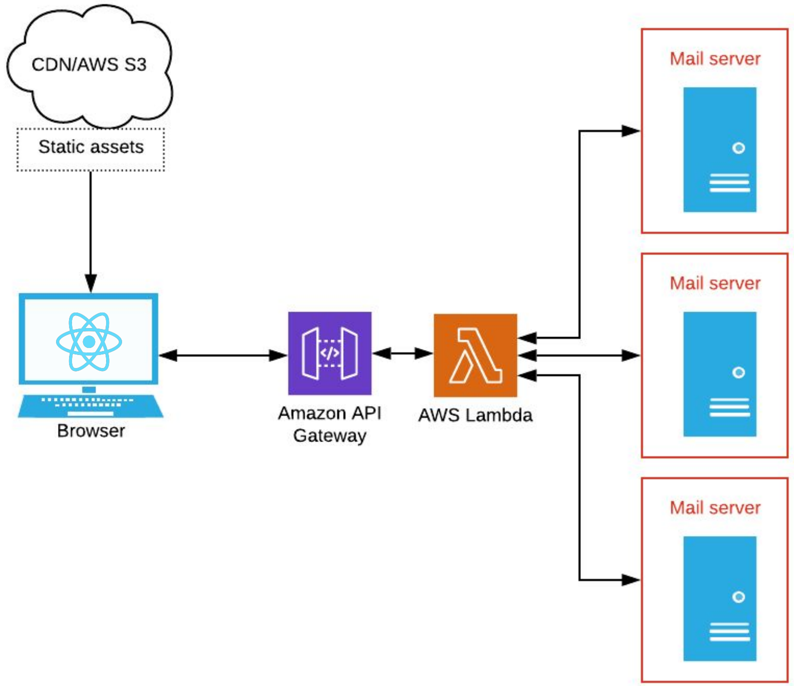
Building the app

- Content-Security-Policy - Reduce attack vectors for XSS
  - `default-src 'self'; img-src: 'self'; style-src https://fonts.googleapis.com 'self'; font-src https://fonts.gstatic.com 'self'`
  - Prevent script embedding in index.html: INLINE_RUNTIME_CHUNK=false npm run build
- CORS
  - Allow (e.g. Access-Control-Allow-Origin)
  - Disallow through CSRF protection - synchronizer token pattern (e.g. csurf in nodejs)
  - “Proxy” setting in package.json
- Strict-Transport-Security and cookies with “secure” and “httpOnly” flags

Limitations

- Auth challenges
  - Stateless: Difficult to store API keys securely in the SPA
    - Local storage vulnerable to XSS
  - Stateful: Manage state on server and use cookies
    - (e.g. express-sessions, cookie-parser in NodeJS) - rfc6265
- Backend and frontend code not isomorphic

Next steps


CDN/AWS S3
Static assets
Browser
Amazon API Gateway
AWS Lambda
Mail server
Mail server
Mail server

Demo

Q&A