Beyond's journey

to server-side rendered application with react.js, graphql, webpack and ...

We’re the UK’s trusted companion for everything after life. Whether you’re looking to sort out your will or you’ve recently lost someone, we’re here to help.

#1 Problem

What technologies should we choose?

CODE

es6+

FRAMEWORKS

react.js

CSS

css modules

BUNDLERS

browserify

API

graphql

API CLIENTS

lokka

STATE MANAGEMENT

redux

REDUX SIDE EFFECTS

redux-observable

CSS PROCESSING

sass/scss

ROUTING

react-mini-router

FORMS

formik

TESTING

sinon

Why 🤔

Just use `create-react-app` or `next.js`

The stack

- ES6+
- React.js
- CSS Modules / SCSS
- GraphQL / Apollo

- Webpack
- redux / redux-saga
- react-router
- ~~redux-form~~ ⇒ formik

Why not CSS-in-JS?

When the project started, we could pick:

- JSS
- ~~Radium~~ (maintenance status now, no future development)
- Rebass
- ~~CSJS~~ (no development since 2017)
- CSS Modules
- Aphrodite

CSS-in-JS

Past - Present by Max Stoiber @Agent Conf

- `Nov 14:` JSS
- `Nov 14:` @vjeux (conference pres)
- `Jan 15:` ~~Radium~~
- `Feb 15:` Rebass
- `May 15:` CSS Modules (not technically CSS-in-JS)
- `Sep 15:` ~~CSSJS~~
- `Oct 15:` Aphrodite
- `Jun 16:` Fela
- `Jul 16:` ~~Glamor~~
- `Oct 16:` jsxstyle
- `Oct 16:` styled-components
- `Dec 16:` styletron
- `Dec 16:` styled-jsx
- `Mar 17:` Astroturf
- `Apr 17:` ~~Glamorous~~
- `May 17:` styled-components v2
- `Jul 17:` Emotion
- `Sep 17:` Linaria
- `Nov 18:` Emotion v10
- `Jun 19:` theme-ui

Potential problems* CSS-in-JS

- watch out for performance issues (#1, #2, #3, #4, #5)
- no cacheable stylesheets (when CSS included in JS)
- more total bytes sent (CSS + runtime lib)
- no async loading of CSS (when CSS included in JS)

When we started 5 years ago...

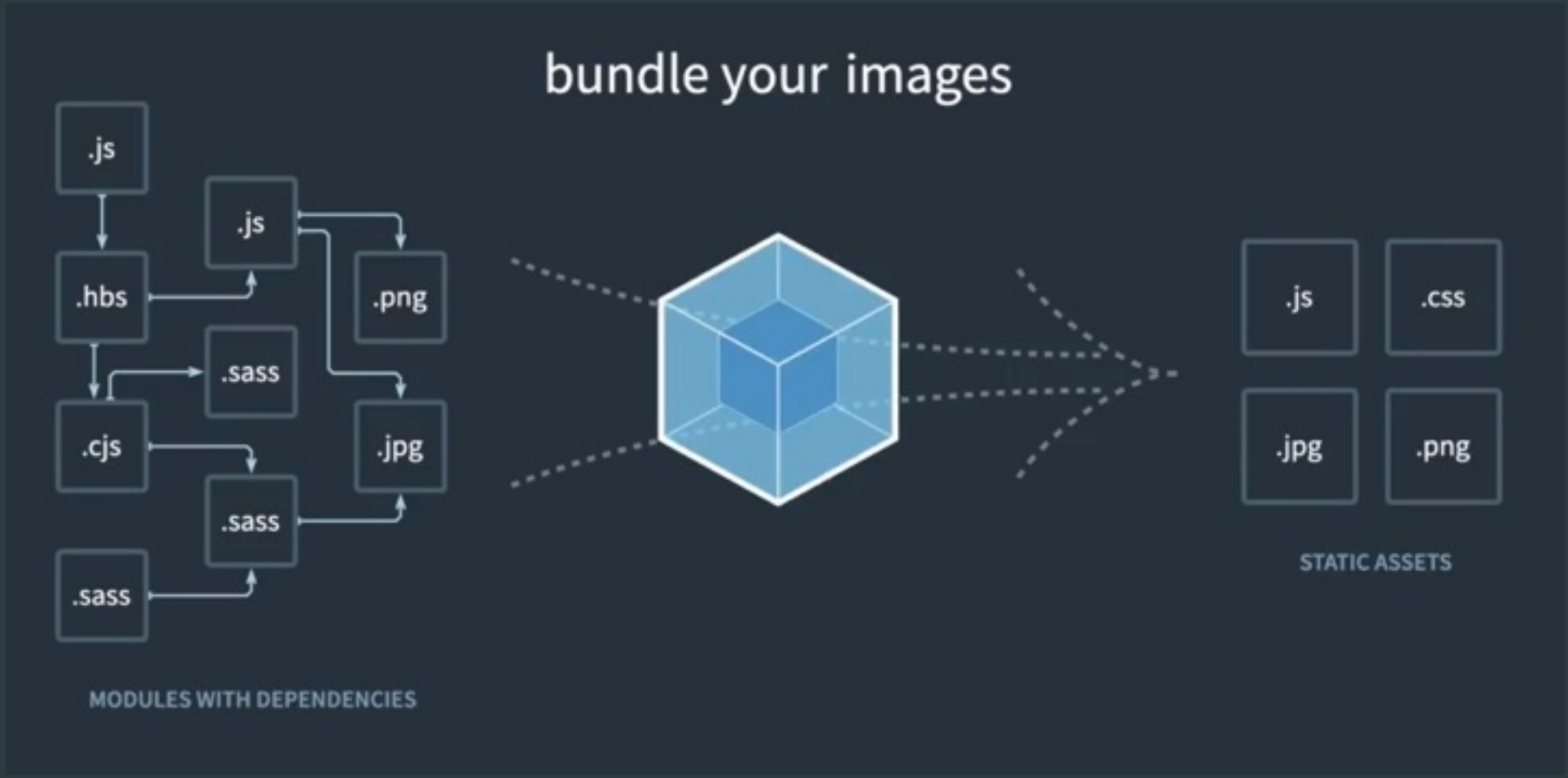
- `create-react-app` & `next.js` didn't exist
- `webpack 2.x` was in beta
- `react-router` 4.x was in beta
- `apollo` 0.x
- `storybook` 2.x
- CSS-in-JS was still super new...

```jsx
const App = () => <h1>Hello World!</h1>;

render(<App />);
```

Before deployment

Build first, `oh my webpack...`


bundle your images
.js
.hbs
.cjs
.sass
.sass
.js
.png
.sass
.jpg
MODULES WITH DEPENDENCIES
.js
.css
.jpg
.png
STATIC ASSETS

#2 Problem

Configuring webpack is hard, really hard...

CSS & Webpack

```
exports.css = ({ sassIncludePaths = [] }, env) => {
  // Shorter classes without path in production
  // Faster rebuilds by not calculating hash while development
  const localIdentName = isProd(env) ? '[local]-[hash:base64:5]' : '[path]---[name]---
  [local]';
  const sourcemapEnabled = false;

  const tests = {
    global: /\.global\.(css|scss)$/,
    local: /^((?!\.global).)*(css|scss)$/,
    css: /\.(css)$/,
  };

  const styleLoader = {
    loader: 'style-loader',
  };

  const miniCssExtractLoader = {
    loader: MiniCssExtractPlugin.loader,
    options: {
      hmr: isDev(env),
    },
```

babel.config.js

```
function createReactCssModulesPlugin(env) {
  // ...code
}

function isSSR(caller) {
  return caller.target === 'node';
}

function isDevLegacy(caller) {
  return caller.devLegacy === true;
}

module.exports = function (api) {
  const development = api.env(['development', 'development-legacy']);
  // if `development-legacy` will be provided, env should be still named as `development`
  const env = development ? 'development' : api.env();
  const ssr = api.caller(isSSR);
  const developmentLegacy = api.caller(isDevLegacy);

  const presets = [
    [
      '@babel/preset-env',
```

WEBPACK IS HARD

If you can, use `create-react-app`, `next.js` etc.

It’s hard to follow

It’s hard to rebuild all optimizations and features that `next.js` or `gatsby` have built in.

SSR & Webpack

We need to build our application twice

1. client (browsers)
2. server (node.js)

Uncharted territory

- hot reloading for client and server
- the same css hashes for client and server
- mock packages that cannot be used on server
- serve production hashed files (.css, .js)
- graphql errors handling

#3 Sometimes works

Development build	❌ Doesn’t work
Production build	✅ Works

react-hot-loader

Tweak React components in real time.

react-toolbox's defaultChecker

```
1  /**
2   * Returns true if the provided element is a component of the provided type.
3   *
4   * @param classType {ReactElement class} - the class of a React Element
5   * @param reactElement {ReactElement} - any React Element (not a real DOM node)
6   */
7  function defaultChecker(classType, reactElement) {
8    if (process.env.NODE_ENV !== 'production') {
9      // https://github.com/gaearon/react-hot-loader/blob/v3.0.0-
       beta.7/docs/Known%20Limitations.md#checking-element-types
10     classType = _react2.default.createElement(classType).type; // eslint-disable-line no-
       param-reassign
11   }
12   return reactElement && reactElement.type === classType;
13 }
14
```

Fix, provide customChecker

```
/*
 * Fix for react-toolbox <Tabs />, <Tab /> components
 * in development mode when react-hot-loader is used
 */
if (process.env.NODE_ENV !== 'production' && process.env.NODE_ENV !== 'test') {
  const {
    overrideComponentTypeChecker,
  } = require('react-toolbox/lib/utils/is-component-of-type');

  overrideComponentTypeChecker(
    (classType, reactElement) =>
      reactElement &&
      (reactElement.type === classType || reactElement.type.name ===
      classType.displayName),
  );
}

```

#4 Problem

DOM mismatch

A bit longer explanation

- React 16 client side render doesn’t update DOM’s style that comes from SSR #11128
- Document that you can’t rely on React 16 SSR patching up differences #10591
- Need a hook for hydration mismatch #11189

#5 Problem

react-helmet & SSR

> *[...] when viewing the HTML of a page, you might see the correct page content with the wrong meta tags. Or the correct page content with empty values for the meta tags.*

The Future of Meta Tag Management for Modern React Development: Introducing React-Helmet-Async by Scott Taylor (Lead Software Engineer at The New York Times)

Solution

 react-helmet-async

#5 Problem

Houston, we have a problem

#6 Problem

chunk loading errors

Deploy with fallback

- on Monday (`a.12zxc.js, b.25asd.js`)
- on Tuesday (`a.12zxc.js, b.98jkl.js, c.45bnm.js`)
- Users from monday doesn’t know about `b.98jkl.js, c.45bnm.js`
- We need to cover that case

Solution

```
class App extends Component {
  componentDidCatch(error, errorInfo) {
    if (!isWebpackChunkLoadingError(error)) {
      sentry.log(error, errorInfo);
      return;
    }

    compareAppVersions().then((result) => {
      if (result) {
        sentry.log(error, { fingerprint: ['webpack', 'error loading chunk'] });

        // According to the docs: https://reactjs.org/docs/react-component.html#componentdidcatch
        //
        // In the event of an error, you can render a fallback UI with componentDidCatch()
        // by calling setState, but this will be deprecated in a future release.
        // Use static getDerivedStateFromError() to handle fallback rendering instead.
        //
        // For now we are okay with that, we will refactor it later.
        this.setState({ error });
        return;
      }

      // We have old application on the client so we ensure that errors are sent to sentry
```

Thanks!

Questions?

Dawid Karabin, Wrocław 2020 - 2021