Making frontend easier and faster by building our own framework
Our Mission was

1. Less Code to Maintain
2. Less Bugs to fix
3. Reduced Bundles
4. Faster Load Times
Our Mission was

The company was

a large online travel agent

with many white label brands

and other websites
Then we saw the code

443 components

972 components * versions
Then we saw the code

12 themes/websites
8+ white labels
Not as easy as we thought...
Circular Dependencies
2500 pug includes
2500 pug includes style imports 5300
Where do we start?
Hello....

- Frontend Team Lead at todopatterson
- Mentor at OpenClassrooms
- Slack Moderator at Treehouse
- Official contributor to the webpack docs
- Former Frontend Architect at bluekiri
- debs_obrien
- Debbie O’Brien
We had to come up with a solution so first we analysed the main problems
Versioning

Wrong versioning

Copy and pasted subfolders

Mixed versions of components in the same release
Styles

Styles out of Control

Themes folders inside components

Unused themes couldn’t be removed

Sites inheriting from other themes
Big Bundles

All components loaded always
And all component versions
Bad Performance
40 minute builds
We had to come up with a plan
Dependencies

Libraries and Vendors handled by a proper dependency management tool
Styles

Separation of styles between projects

Component Styles at component level

No themes inside the Components

Sass maps to modify theme

Add RTL package for left to right websites
JavaScript

Remove logic in pug files

Remove T3 JavaScript framework

Remove TypeScript in favour of JS

All JavaScript at component level
Operations

An application lifecycle to handle:

code sharing and reviews

linting, testing and versioning

packaging, building and releasing
Now that’s what I call a plan
Meet
Alexandria

Our custom frontend framework
Meet Alexandria

Another bloody frontend framework
Meet Alexandria

Yeah...

I know.....
Meet Alexandria

Yeah...

I know.....

But.......
Meet Alexandria

We needed:

a solution that suited our business needs

no extra training for the team
The bosses were happy
Meet Alexandria

Our custom frontend framework

Alexandria targets the proper use of current technologies and simplification when possible.
This is what we achieved
Packages structure

alexandria (rendering, resolution and devtools)

alexandria components
alexandria styles
alexandria icons

brand

business line project
Styles

Manual Style Imports

Manual tedious task
Many problems - duplicate styles
Difficult to maintain

220 style imports to maintain
Styles

Manual Style Imports

Manual tedious task
Many problems - duplicate styles
Difficult to maintain

220 style imports to maintain

Automatic Style Imports

All styles rendered from model
Webpack bundles and minifies
Post CSS removes any duplicates

0 style imports to maintain
Styles

Sass Variables

Long variable names
Difficult to understand
Making changes was time consuming

Sass Maps

Short names
Easy to understand
Making changes was fast
Icons

Icon Package

**Embedded svg** icons
Less **Css** code
Easier to position
Better **Css Control**
Better **accessibility**
Quicker builds
Less server **requests**
No more **icon problems**
Icons

Icon Package

Embedded svg icons
Less CSS code
Easier to position
Better CSS Control
Better accessibility
Quicker builds
Less server requests
No more icon problems
Icons

Icon Package

**Embedded svg** icons
**Less Css** code
**Easier** to position
**Better** Css Control
Better **accessibility**
**Quicker builds**
**Less** server **requests**
**No more** icon problems
Components

Component Structure

Components are **self-contained**

Each component has a main `index.js` file to **render** its **pug**

All **client-side code** [CSS and browser JS] is also **inside** the **component**

**No themes folders**
Components

Component Package

**Mini** components
**Less** CSS duplicates and conflicts
**Quicker** to implement and maintain
Styles modified at **brand level**
Modifications **only affect your brand**
Better **control**
Full **testing** for these components
**Consistency** across all pages
Components

Component Development

View components:
with npm script
direct in project
with real data
different models
individual component or a layout
all components with variations

Quicker rendering - quicker development
Components

Component Development

View components:
with npm script
direct in project
with real data
different models
individual component or a layout
all components with variations

Quicker rendering - quicker development
Versioning

Wrong Versioning
Copy and pasted code into a new folder called v2
v3 was a copy of which version?

Semantic Versioning
Alexandria code is properly versioned with npm and semantic versioning

1.2.5
major minor patch
Builds and Performance

Bundle size was

557 kb

> 250kb recommended max size
Builds and Performance

Bundle size was

557 kb

> 250kb recommended max size

Bundle size now

139 kb

< 250kb recommended max size

70-80% weight loss
Builds and Performance

Server Requests was

53

CSS and JS requests
Builds and Performance

Server Requests was

53

CSS and JS requests

Server requests now

4

massive reduction

CSS and JS requests
Builds and Performance

Build Time was

40 mins
Builds and Performance

Build Time was 40 mins

Build Time now 3 mins

massive reduction
Builds and Performance

Page load was 10 secs
## Builds and Performance

<table>
<thead>
<tr>
<th>Page load was</th>
<th>Page load now</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 secs</td>
<td>1.5 secs</td>
</tr>
</tbody>
</table>
What we achieved

1. Less Code to Maintain
2. Less Bugs to fix
3. Reduced Bundles
4. Faster Load Times
5. Better SEO results
And so Alexandria lived happily ever after
Thanks

I know that was a lot